Enriched Category Basics

Eugene W. Stark

Department of Computer Science
Stony Brook University
Stony Brook, New York 11794 USA

June 17, 2024

Abstract

The notion of an enriched category generalizes the concept of category by replacing the hom-sets of an ordinary category by objects of an arbitrary monoidal category. In this article we give a formal definition of enriched categories and we give formal proofs of a relatively narrow selection of facts about them. One of the main results is a proof that a closed monoidal category can be regarded as a category "enriched in itself". The other main result is a proof of a version of the Yoneda Lemma for enriched categories.

Contents

Introduction

The notion of an enriched category [1] generalizes the concept of category by replacing the hom-sets of an ordinary category by objects of an arbitrary monoidal category $\mathcal{V}$. The choice, for each object a, of a distinguished element $id\ a : a \to a$ as an identity, is replaced by an arrow $Id\ a : \mathcal{I} \to Hom\ a\ a$ of $\mathcal{V}$. The composition operation is similarly replaced by a family of arrows $Comp\ a\ b\ c : Hom\ B\ C \otimes Hom\ A\ B \to Hom\ A\ C$ of $\mathcal{V}$. The identity and composition are required to satisfy unit and associativity laws which are expressed as commutative diagrams in $\mathcal{V}$. Of particular interest is the case in which $\mathcal{V}$ is symmetric monoidal and closed; in that case, as Kelly states ([1], Section 1.6): "The structure of $\mathcal{V}$-**CAT** then becomes rich enough to permit of Yoneda-lemma arguments formally identical with those in **CAT**."

The goal of this article is to formalize the basic definition of enriched category and some related notions, and to prove a relatively narrow selection of facts about these definitions. For reference and inspiration, we follow the early sections of the book by Kelly [1]; however a comprehensive formalization of the material in that book is explicitly not our objective here. Rather, beyond the basic definitions we are primarily interested in the following two results: (1) that a closed monoidal category can be regarded as a category "enriched in itself"; and (2) the Yoneda Lemma for enriched categories (specifically, the weak form considered in Section 1.9 of [1]). We needed the basic definitions and result (1) for use in a separate article [4]. Although this material could have been included as part of that other article, as it is general material that does not depend on the specific application considered there, it seemed best to present it as a stand-alone development that would be more readily accessible for use by others. As far as result (2) is concerned, we originally formalized and proved it as part of our exploration leading up to [4]. Ultimately, we did not find result (2) to be necessary for the satisfactory development of that work, but as it is a result of general interest whose formalization did involve some struggle to achieve, it seems worthwhile to include it here.

This article is organized as follows: In Chapter 1 we give formal definitions for the notions "closed monoidal category" and "cartesian closed monoidal category" and prove some facts about them. This builds on the formal development of the theory of monoidal categories in our previous article [3]. The main goals of this section are to prove some general facts about exponentials that are used in [4], and to do most of the preliminary work (the parts that do not specifically depend on the definition of enriched category) involved in showing that a closed monoidal category is "enriched in itself". In Chapter 2 we give definitions for "enriched category" and the related notions "enriched functor," "enriched natural transformation," and "underlying category," and we complete the formal statement and proof of "self-enrichment." We then continue with the definition of the opposite of an enriched category, give definitions for the notions of covariant and contravariant enriched hom functors, and prove corresponding covariant and contravariant versions of the Yoneda Lemma.

Chapter 1

Closed Monoidal Categories

A *closed monoidal category* is a monoidal category such that for every object b, the functor - $\otimes$ b is a left adjoint functor. A right adjoint to this functor takes each object c to the *exponential* $exp\ b\ c$. The adjunction yields a natural bijection between $hom\ (- \otimes b)\ c$ and $hom\ -\ (exp\ b\ c)$. In enriched category theory, the notion of "hom-set" from classical category theory is generalized to that of "hom-object" in a monoidal category. When the monoidal category in question is closed, much of the theory of set-based categories can be reproduced in the more general enriched setting. The main purpose of this section is to prepare the way for such a development; in particular we do the main work required to show that a closed monoidal category is "enriched in itself."

theory *ClosedMonoidalCategory*
imports *MonoidalCategory.CartesianMonoidalCategory*
begin

1.1 Definition and Basic Facts

As is pointed out in [2], unless symmetry is assumed as part of the definition, there are in fact two notions of closed monoidal category: *left*-closed monoidal category and *right*-closed monoidal category. Here we define versions with and without symmetry, so that we can identify the places where symmetry is actually required.

locale *closed-monoidal-category* =
monoidal-category +
assumes *left-adjoint-tensor*: $\bigwedge b.\ ide\ b \Longrightarrow left\text{-}adjoint\text{-}functor\ C\ C\ (\lambda x.\ x \otimes b)$

locale *closed-symmetric-monoidal-category* =
closed-monoidal-category +
symmetric-monoidal-category

Similarly to what we have done in previous work, besides the definition of *closed-monoidal-category*, which adds an assumed property to *monoidal-category* but not any additional structure, we find it convenient also to define *elementary-closed-monoidal-category*, which assumes particular exponential structure to have been chosen, and uses this given structure to express the properties of a closed monoidal category in a more elementary way.

locale *elementary-closed-monoidal-category* =
  monoidal-category +
fixes *exp* :: 'a ⇒ 'a ⇒ 'a
and *eval* :: 'a ⇒ 'a ⇒ 'a
and *Curry* :: 'a ⇒ 'a ⇒ 'a ⇒ 'a ⇒ 'a
assumes *eval-in-hom-ax*: ⟦ *ide b*; *ide c* ⟧ ⟹ «*eval b c* : *exp b c* ⊗ *b* → *c*»
and *ide-exp* [*intro*, *simp*]: ⟦ *ide b*; *ide c* ⟧ ⟹ *ide* (*exp b c*)
and *Curry-in-hom-ax*: ⟦ *ide a*; *ide b*; *ide c*; «*g* : *a* ⊗ *b* → *c*» ⟧
      ⟹ «*Curry a b c g* : *a* → *exp b c*»
and *Uncurry-Curry*: ⟦ *ide a*; *ide b*; *ide c*; «*g* : *a* ⊗ *b* → *c*» ⟧
      ⟹ *eval b c* · (*Curry a b c g* ⊗ *b*) = *g*
and *Curry-Uncurry*: ⟦ *ide a*; *ide b*; *ide c*; «*h* : *a* → *exp b c*» ⟧
      ⟹ *Curry a b c* (*eval b c* · (*h* ⊗ *b*)) = *h*

locale *elementary-closed-symmetric-monoidal-category* =
  symmetric-monoidal-category +
  elementary-closed-monoidal-category
begin

  sublocale *elementary-symmetric-monoidal-category*
      C tensor $\mathcal{I}$ *lunit runit assoc sym*
    using *induces-elementary-symmetric-monoidal-category*$_{CMC}$ **by** *blast*

end

We now show that, except for the fact that a particular choice of structure has been made, closed monoidal categories and elementary closed monoidal categories amount to the same thing.

1.1.1 An ECMC is a CMC

context *elementary-closed-monoidal-category*
begin

  notation *Curry* (*Curry*[-, -, -])

  abbreviation *Uncurry* (*Uncurry*[-, -])
  where *Uncurry*[*b*, *c*] *f* ≡ *eval b c* · (*f* ⊗ *b*)

  lemma *Curry-in-hom* [*intro*]:
  assumes *ide a* **and** *ide b* **and** «*g* : *a* ⊗ *b* → *c*» **and** *y* = *exp b c*
  shows «*Curry*[*a*, *b*, *c*] *g* : *a* → *y*»
    using *assms Curry-in-hom-ax* [*of a b c g*] **by** *fastforce*

lemma *Curry-simps* [*simp*]:
assumes *ide a* **and** *ide b* **and** «*g* : *a* ⊗ *b* → *c*»
shows *arr* (*Curry*[*a*, *b*, *c*] *g*)
and *dom* (*Curry*[*a*, *b*, *c*] *g*) = *a*
and *cod* (*Curry*[*a*, *b*, *c*] *g*) = *exp b c*
  using *assms Curry-in-hom* **by** *blast*+

lemma *eval-in-hom*$_{ECMC}$ [*intro*]:
assumes *ide b* **and** *ide c* **and** *x* = *exp b c* ⊗ *b*
shows «*eval b c* : *x* → *c*»
  using *assms eval-in-hom-ax* **by** *blast*

lemma *eval-simps* [*simp*]:
assumes *ide b* **and** *ide c*
shows *arr* (*eval b c*) **and** *dom* (*eval b c*) = *exp b c* ⊗ *b* **and** *cod* (*eval b c*) = *c*
  using *assms eval-in-hom*$_{ECMC}$ **by** *blast*+

lemma *Uncurry-in-hom* [*intro*]:
assumes *ide b* **and** *ide c* **and** «*f* : *a* → *exp b c*» **and** *x* = *a* ⊗ *b*
shows «*Uncurry*[*b*, *c*] *f* : *x* → *c*»
  using *assms* **by** *auto*

lemma *Uncurry-simps* [*simp*]:
assumes *ide b* **and** *ide c* **and** «*f* : *a* → *exp b c*»
shows *arr* (*Uncurry*[*b*, *c*] *f*)
and *dom* (*Uncurry*[*b*, *c*] *f*) = *a* ⊗ *b*
and *cod* (*Uncurry*[*b*, *c*] *f*) = *c*
  using *assms Uncurry-in-hom* **by** *blast*+

lemma *Uncurry-exp*:
assumes *ide a* **and** *ide b*
shows *Uncurry*[*a*, *b*] (*exp a b*) = *eval a b*
  using *assms*
  by (*metis comp-arr-dom eval-in-hom*$_{ECMC}$ *in-homE*)

lemma *comp-Curry-arr*:
assumes *ide b* **and** «*f* : *x* → *a*» **and** «*g* : *a* ⊗ *b* → *c*»
shows *Curry*[*a*, *b*, *c*] *g* · *f* = *Curry*[*x*, *b*, *c*] (*g* · (*f* ⊗ *b*))
proof −
  have *a*: *ide a* **and** *c*: *ide c* **and** *x*: *ide x*
    using *assms*(*2*−*3*) **by** *auto*
  have *Curry*[*a*, *b*, *c*] *g* · *f* =
        Curry[*x*, *b*, *c*] (*Uncurry*[*b*, *c*] (*Curry*[*a*, *b*, *c*] *g* · *f*))
    using *assms*(*1*−*3*) *a c x Curry-Uncurry comp-in-homI Curry-in-hom*
    by *presburger*
  also have ... = *Curry*[*x*, *b*, *c*] (*eval b c* · (*Curry*[*a*, *b*, *c*] *g* ⊗ *b*) · (*f* ⊗ *b*))
    using *assms a c interchange*
    by (*metis comp-ide-self Curry-in-hom ideD*(*1*) *seqI*′)
  also have ... = *Curry*[*x*, *b*, *c*] (*Uncurry*[*b*, *c*] (*Curry*[*a*, *b*, *c*] *g*) · (*f* ⊗ *b*))

```
    using comp-assoc by simp
  also have ... = Curry[x, b, c] (g · (f ⊗ b))
    using a c assms(1,3) Uncurry-Curry by simp
  finally show ?thesis by blast
qed

lemma terminal-arrow-from-functor-eval:
assumes ide b and ide c
shows terminal-arrow-from-functor C C (λx. T (x, b)) (exp b c) c (eval b c)
proof −
  interpret functor C C ‹λx. T (x, b)›
    using assms(1) interchange T.is-extensional
    by unfold-locales auto
  interpret arrow-from-functor C C ‹λx. T (x, b)› ‹exp b c› c ‹eval b c›
    using assms eval-in-hom_ECMC
    by unfold-locales auto
  show ?thesis
  proof
    show ⋀a f. arrow-from-functor C C (λx. T (x, b)) a c f ⟹
              ∃!g. arrow-from-functor.is-coext C C
                     (λx. T (x, b)) (exp b c) (eval b c) a f g
    proof −
      fix a f
      assume f: arrow-from-functor C C (λx. T (x, b)) a c f
      interpret f: arrow-from-functor C C ‹λx. T (x, b)› a c f
        using f by simp
      show ∃!g. is-coext a f g
      proof
        have a: ide a
          using f.arrow by simp
        show is-coext a f (Curry[a, b, c] f)
          unfolding is-coext-def
          using assms a Curry-in-hom Uncurry-Curry f.arrow by force
        show ⋀g. is-coext a f g ⟹ g = Curry[a, b, c] f
          unfolding is-coext-def
          using assms a Curry-Uncurry f.arrow arrI by force
      qed
    qed
  qed
qed

lemma is-closed-monoidal-category:
shows closed-monoidal-category C T α ι
  using T.is-extensional interchange terminal-arrow-from-functor-eval
  apply unfold-locales
     apply auto[5]
  by metis

lemma retraction-eval-ide-self:
```

assumes *ide a*
shows *retraction* (*eval a a*)
by (*metis Uncurry-Curry assms comp-lunit-lunit'*(*1*) *ide-unity comp-assoc lunit-in-hom retractionI*)

end

context *elementary-closed-symmetric-monoidal-category*
begin

lemma *is-closed-symmetric-monoidal-category*:
shows *closed-symmetric-monoidal-category* $C\ T\ \alpha\ \iota\ \sigma$
by (*simp add*: *closed-symmetric-monoidal-category.intro is-closed-monoidal-category symmetric-monoidal-category-axioms*)

end

1.1.2 A CMC Extends to an ECMC

context *closed-monoidal-category*
begin

lemma *has-exponentials*:
assumes *ide b* **and** *ide c*
shows $\exists\, x\ e.\ ide\ x \wedge «e : x \otimes b \rightarrow c» \wedge$
$(\forall\, a\ g.\ ide\ a \wedge$
$«g : a \otimes b \rightarrow c» \longrightarrow (\exists!f.\ «f : a \rightarrow x» \wedge g = e \cdot (f \otimes b)))$
proof –
interpret *F*: *left-adjoint-functor C C* ‹$\lambda x.\ x \otimes b$›
using *assms*(*1*) *left-adjoint-tensor* **by** *simp*
obtain *x e* **where** *e*: *terminal-arrow-from-functor C C* ($\lambda x.\ x \otimes b$) *x c e*
using *assms F.ex-terminal-arrow* [*of c*] **by** *auto*
interpret *e*: *terminal-arrow-from-functor C C* ‹$\lambda x.\ x \otimes b$› *x c e*
using *e* **by** *simp*
have $\bigwedge a\ g.\ [\![\ ide\ a;\ «g : a \otimes b \rightarrow c»\ ]\!]$
$\Longrightarrow \exists!f.\ «f : a \rightarrow x» \wedge g = e \cdot (f \otimes b)$
using *e.is-terminal category-axioms F.functor-axioms*
unfolding *e.is-coext-def arrow-from-functor-def arrow-from-functor-axioms-def*
by *simp*
thus *?thesis*
using *e.arrow* **by** *metis*
qed

definition *some-exp* ($exp^?$)
where $exp^?\ b\ c \equiv SOME\ x.\ ide\ x \wedge$
$(\exists\, e.\ «e : x \otimes b \rightarrow c» \wedge$

$(\forall\, a\ g.\ ide\ a \wedge «g : a \otimes b \to c»$
$\longrightarrow (\exists!f.\ «f : a \to x» \wedge g = e \cdot (f \otimes b))))$

definition *some-eval* ($eval^?$)
where $eval^?\ b\ c \equiv SOME\ e.\ «e : exp^?\ b\ c \otimes b \to c» \wedge$
$(\forall\, a\ g.\ ide\ a \wedge «g : a \otimes b \to c»$
$\longrightarrow (\exists!f.\ «f : a \to exp^?\ b\ c» \wedge g = e \cdot (f \otimes b)))$

definition *some-Curry* ($Curry^?$[-, -, -])
where $Curry^?[a, b, c]\ g \equiv$
$THE\ f.\ «f : a \to exp^?\ b\ c» \wedge g = eval^?\ b\ c \cdot (f \otimes b)$

abbreviation *some-Uncurry* ($Uncurry^?$[-, -])
where $Uncurry^?[b, c]\ f \equiv eval^?\ b\ c \cdot (f \otimes b)$

lemma *Curry-uniqueness*:
assumes *ide b* **and** *ide c*
shows $ide\ (exp^?\ b\ c)$ **and** $«eval^?\ b\ c : exp^?\ b\ c \otimes b \to c»$
and $[\![\ ide\ a;\ «g : a \otimes b \to c»\ ]\!]$
$\Longrightarrow \exists!f.\ «f : a \to exp^?\ b\ c» \wedge g = Uncurry^?[b, c]\ f$
  using *assms some-exp-def some-eval-def has-exponentials*
    $someI\text{-}ex\ [of\ \lambda x.\ ide\ x \wedge (\exists\, e.\ «e : x \otimes b \to c» \wedge$
    $(\forall\, a\ g.\ ide\ a \wedge «g : a \otimes b \to c»$
    $\longrightarrow (\exists!f.\ «f : a \to x» \wedge g = e \cdot (f \otimes b))))]$
    $someI\text{-}ex\ [of\ \lambda e.\ «e : exp^?\ b\ c \otimes b \to c» \wedge$
    $(\forall\, a\ g.\ ide\ a \wedge «g : a \otimes b \to c»$
    $\longrightarrow (\exists!f.\ «f : a \to exp^?\ b\ c» \wedge g = e \cdot (f \otimes b)))]$
  by *auto*

lemma *some-eval-in-hom* [*intro*]:
assumes *ide b* **and** *ide c* **and** $x = exp^?\ b\ c \otimes b$
shows $«eval^?\ b\ c : x \to c»$
  using *assms Curry-uniqueness* **by** *simp*

lemma *some-Uncurry-some-Curry*:
assumes *ide a* **and** *ide b* **and** $«g : a \otimes b \to c»$
shows $«Curry^?[a, b, c]\ g : a \to exp^?\ b\ c»$
and $Uncurry^?[b, c]\ (Curry^?[a, b, c]\ g) = g$
proof −
  have *ide c*
    using *assms*(*3*) **by** *auto*
  hence *1*: $«Curry^?[a, b, c]\ g : a \to exp^?\ b\ c» \wedge$
    $g = Uncurry^?[b, c]\ (Curry^?[a, b, c]\ g)$
    using *assms some-Curry-def Curry-uniqueness*
      $theI'\ [of\ \lambda f.\ «f : a \to exp^?\ b\ c» \wedge g = Uncurry^?[b, c]\ f]$
    by *simp*
  show $«Curry^?[a, b, c]\ g : a \to exp^?\ b\ c»$
    using *1* **by** *simp*
  show $Uncurry^?[b, c]\ (Curry^?[a, b, c]\ g) = g$

using *1* **by** *simp*
qed

lemma *some-Curry-some-Uncurry*:
assumes *ide b* **and** *ide c* **and** «$h : a \rightarrow exp^? \ b \ c$»
shows $Curry^?[a, b, c] \ (Uncurry^?[b, c] \ h) = h$
proof –
have $\exists!f.$ «$f : a \rightarrow exp^? \ b \ c$» $\wedge$ $Uncurry^?[b, c] \ h = Uncurry^?[b, c] \ f$
using *assms ide-dom ide-in-hom*
Curry-uniqueness(*3*) [*of b c a* $Uncurry^?[b, c] \ h$]
by *auto*
moreover have «$h : a \rightarrow exp^? \ b \ c$» $\wedge$ $Uncurry^?[b, c] \ h = Uncurry^?[b, c] \ h$
using *assms* **by** *simp*
ultimately show *?thesis*
using *assms some-Curry-def Curry-uniqueness some-Uncurry-some-Curry*
the1-equality [*of* $\lambda f.$ «$f : a \rightarrow$ *some-exp b c*» $\wedge$
$Uncurry^?[b, c] \ h = Uncurry^?[b, c] \ f$]
by *simp*
qed

lemma *extends-to-elementary-closed-monoidal-category*$_{CMC}$:
shows *elementary-closed-monoidal-category*
C T α ι *some-exp some-eval some-Curry*
using *Curry-uniqueness some-Uncurry-some-Curry*
some-Curry-some-Uncurry
by *unfold-locales auto*

end

context *closed-symmetric-monoidal-category*
begin

lemma *extends-to-elementary-closed-symmetric-monoidal-category*$_{CMC}$:
shows *elementary-closed-symmetric-monoidal-category*
C T α ι σ *some-exp some-eval some-Curry*
by (*simp add*: *elementary-closed-symmetric-monoidal-category-def*
extends-to-elementary-closed-monoidal-category$_{CMC}$
symmetric-monoidal-category-axioms)

end

1.2 Internal Hom Functors

For each object x of a closed monoidal category C, we can define a covariant endofunctor $Exp^{\rightarrow} \ x \ -$ of C, which takes each arrow g to an arrow «$Exp^{\rightarrow} \ x \ g : exp \ x \ (dom \ g) \rightarrow exp \ x \ (cod \ g)$». Similarly, for each object y, we can define a contravariant endofunctor $Exp^{\leftarrow} \ - \ y$ of C, which takes each arrow f of C^{op} to an arrow «$Exp^{\leftarrow} \ f \ y : exp \ (cod \ f) \ y \rightarrow exp \ (dom \ f) \ y$» of C.

These two endofunctors commute with each other and compose to form a single binary "internal hom" functor *Exp* from $C^{op} \times C$ to C.

context *elementary-closed-monoidal-category*
begin

abbreviation *cov-Exp* ($Exp^{\rightarrow}$)
where $Exp^{\rightarrow}\ x\ g \equiv$ *if arr g*
then $Curry[exp\ x\ (dom\ g),\ x,\ cod\ g]\ (g \cdot eval\ x\ (dom\ g))$
else null

abbreviation *cnt-Exp* ($Exp^{\leftarrow}$)
where $Exp^{\leftarrow}\ f\ y \equiv$ *if arr f*
then $Curry[exp\ (cod\ f)\ y,\ dom\ f,\ y]$
$(eval\ (cod\ f)\ y \cdot (exp\ (cod\ f)\ y \otimes f))$
else null

lemma *cov-Exp-in-hom*:
assumes *ide x* **and** *arr g*
shows «$Exp^{\rightarrow}\ x\ g : exp\ x\ (dom\ g) \rightarrow exp\ x\ (cod\ g)$»
using *assms* **by** *auto*

lemma *cnt-Exp-in-hom*:
assumes *arr f* **and** *ide y*
shows «$Exp^{\leftarrow}\ f\ y : exp\ (cod\ f)\ y \rightarrow exp\ (dom\ f)\ y$»
using *assms* **by** *force*

lemma *cov-Exp-ide*:
assumes *ide a* **and** *ide b*
shows $Exp^{\rightarrow}\ a\ b = exp\ a\ b$
using *assms*
by (*metis comp-ide-arr Curry-Uncurry* $eval\text{-}in\text{-}hom_{ECMC}$ *ideD*(*2*−*3*) *ide-exp ide-in-hom seqI′ Uncurry-exp*)

lemma *cnt-Exp-ide*:
assumes *ide a* **and** *ide b*
shows $Exp^{\leftarrow}\ a\ b = exp\ a\ b$
using *assms Curry-Uncurry ide-exp ide-in-hom* **by** *force*

lemma *cov-Exp-comp*:
assumes *ide x* **and** *seq g f*
shows $Exp^{\rightarrow}\ x\ (g \cdot f) = Exp^{\rightarrow}\ x\ g \cdot Exp^{\rightarrow}\ x\ f$
proof −
have $Exp^{\rightarrow}\ x\ g \cdot Exp^{\rightarrow}\ x\ f =$
$Curry[exp\ x\ (cod\ f),\ x,\ cod\ g]\ (g \cdot eval\ x\ (cod\ f)) \cdot$
$Curry[exp\ x\ (dom\ f),\ x,\ cod\ f]\ (f \cdot eval\ x\ (dom\ f))$
using *assms* **by** *auto*
also have ... $= Curry[exp\ x\ (dom\ f),\ x,\ cod\ g]$
$((g \cdot eval\ x\ (dom\ g)) \cdot$
$(Curry[exp\ x\ (dom\ f),\ x,\ cod\ f]\ (f \cdot eval\ x\ (dom\ f)) \otimes x))$

using *assms cov-Exp-in-hom comp-Curry-arr* **by** *auto*
also have ... = $Exp^{\rightarrow}$ *x* (*g* · *f*)
using *assms Uncurry-Curry comp-assoc* **by** *fastforce*
finally show *?thesis* **by** *simp*
qed

lemma *cnt-Exp-comp*:
assumes *seq g f* **and** *ide y*
shows $Exp^{\leftarrow}$ (*g* · *f*) *y* = $Exp^{\leftarrow}$ *f y* · $Exp^{\leftarrow}$ *g y*
proof −
have $Exp^{\leftarrow}$ *f y* · $Exp^{\leftarrow}$ *g y* =
Curry[*exp* (*cod g*) *y*, *dom f*, *y*]
((*eval* (*cod f*) *y* · (*exp* (*cod f*) *y* ⊗ *f*)) ·
(*Curry*[*exp* (*cod g*) *y*, *cod f*, *y*]
(*eval* (*cod g*) *y* · (*exp* (*cod g*) *y* ⊗ *g*)) ⊗ *dom f*))
using *assms*
comp-Curry-arr
[*of dom f Curry*[*exp* (*cod g*) *y*, *cod f*, *y*]
(*eval* (*cod g*) *y* · (*exp* (*cod g*) *y* ⊗ *g*))]
by *fastforce*
also have ... = *Curry*[*exp* (*cod g*) *y*, *dom f*, *y*]
((*Uncurry*[*cod f*, *y*]
(*Curry*[*exp* (*cod g*) *y*, *cod f*, *y*]
(*eval* (*cod g*) *y* · (*exp* (*cod g*) *y* ⊗ *g*)))) ·
(*exp* (*cod g*) *y* ⊗ *f*))
using *assms interchange comp-arr-dom comp-cod-arr comp-assoc* **by** *auto*
also have ... = *Curry*[*exp* (*cod g*) *y*, *dom f*, *y*]
((*eval* (*cod g*) *y* · (*exp* (*cod g*) *y* ⊗ *g*)) · (*exp* (*cod g*) *y* ⊗ *f*))
using *assms Uncurry-Curry* **by** *auto*
also have ... = $Exp^{\leftarrow}$ (*g* · *f*) *y*
using *assms interchange comp-assoc* **by** *auto*
finally show *?thesis* **by** *simp*
qed

lemma *functor-cov-Exp*:
assumes *ide x*
shows *functor C C* ($Exp^{\rightarrow}$ *x*)
using *assms cov-Exp-ide cov-Exp-in-hom cov-Exp-comp*
by *unfold-locales auto*

interpretation *Cop*: *dual-category C* **..**

lemma *functor-cnt-Exp*:
assumes *ide x*
shows *functor Cop.comp C* (λ*f*. $Exp^{\leftarrow}$ *f x*)
using *assms cnt-Exp-ide cnt-Exp-in-hom cnt-Exp-comp*
by *unfold-locales auto*

lemma *cov-cnt-Exp-commute*:

```
assumes arr f and arr g
shows Exp→ (dom f) g · Exp← f (dom g) =
      Exp← f (cod g) · Exp→ (cod f) g
proof –
  have Exp→ (dom f) g · Exp← f (dom g) =
        Curry[exp (cod f) (dom g), dom f, cod g]
          ((g · eval (dom f) (dom g)) ·
             (Curry[exp (cod f) (dom g), dom f, dom g]
                (eval (cod f) (dom g) · (exp (cod f) (dom g) ⊗ f)) ⊗ dom f))
    using assms cnt-Exp-in-hom comp-Curry-arr by force
  also have ... = Curry[exp (cod f) (dom g), dom f, cod g]
                    (Uncurry[cod f, cod g] (Exp→ (cod f) g) ·
                       (exp (cod f) (dom g) ⊗ f))
    using assms comp-assoc Uncurry-Curry by auto
  also have ... = Curry[exp (cod f) (dom g), dom f, cod g]
                    (eval (cod f) (cod g) · (Exp→ (cod f) g ⊗ cod f) ·
                       (exp (cod f) (dom g) ⊗ f))
    using comp-assoc by auto
  also have ... = Curry[exp (cod f) (dom g), dom f, cod g]
                    (eval (cod f) (cod g) · (Exp→ (cod f) g ⊗ f))
    using assms interchange comp-arr-dom comp-cod-arr
    by (metis cov-Exp-in-hom ide-cod in-homE)
  also have ... = Curry[exp (cod f) (dom g), dom f, cod g]
                    (eval (cod f) (cod g) ·
                       (exp (cod f) (cod g) ⊗ f) · (Exp→ (cod f) g ⊗ dom f))
    using assms interchange comp-arr-dom comp-cod-arr cov-Exp-in-hom
    by auto
  also have ... = Exp← f (cod g) · Exp→ (cod f) g
    using assms cov-Exp-in-hom comp-assoc
          comp-Curry-arr
            [of dom f Exp→ (cod f) g exp (cod f) (dom g) -
                eval (cod f) (cod g) · (exp (cod f) (cod g) ⊗ f) cod g]
    by simp
  finally show ?thesis by simp
qed

definition Exp
where Exp f g ≡ Exp→ (dom f) g · Exp← f (dom g)

lemma Exp-in-hom:
assumes arr f and arr g
shows «Exp f g : Exp (cod f) (dom g) → Exp (dom f) (cod g)»
  using Exp-def assms(1−2) cnt-Exp-ide cov-Exp-ide by auto

lemma Exp-ide:
assumes ide a and ide b
shows Exp a b = exp a b
  unfolding Exp-def
  using assms cov-Exp-ide cnt-Exp-ide by simp
```

lemma *Exp-comp*:
assumes *seq g f* **and** *seq k h*
shows *Exp* $(g \cdot f)$ $(k \cdot h)$ = *Exp f k* · *Exp g h*
proof –
  have *Exp* $(g \cdot f)$ $(k \cdot h)$ = $Exp^{\rightarrow}$ (*dom f*) $(k \cdot h)$ · $Exp^{\leftarrow}$ $(g \cdot f)$ (*dom h*)
    unfolding *Exp-def*
    using *assms* **by** *auto*
  also have ... = ($Exp^{\rightarrow}$ (*dom f*) *k* · $Exp^{\rightarrow}$ (*dom f*) *h*) ·
                   ($Exp^{\leftarrow}$ *f* (*dom h*) · $Exp^{\leftarrow}$ *g* (*dom h*))
    using *assms cov-Exp-comp cnt-Exp-comp* **by** *auto*
  also have ... = ($Exp^{\rightarrow}$ (*dom f*) *k* · $Exp^{\leftarrow}$ *f* (*dom k*)) ·
                   ($Exp^{\rightarrow}$ (*dom g*) *h* · $Exp^{\leftarrow}$ *g* (*dom h*))
    using *assms comp-assoc cov-cnt-Exp-commute*
    by (*metis* (*no-types*, *lifting*) *seqE*)
  also have ... = *Exp f k* · *Exp g h*
    unfolding *Exp-def* **by** *blast*
  finally show *?thesis* **by** *blast*
qed

interpretation *CopxC*: *product-category Cop.comp C* **..**

lemma *functor-Exp*:
shows *binary-functor Cop.comp C C* (*λfg. Exp* (*fst fg*) (*snd fg*))
  using *Exp-in-hom*
  apply *unfold-locales*
      apply *auto*[*4*]
  using *Exp-def*
    apply *auto*[*2*]
  using *Exp-comp*
  by *fastforce*

lemma *Exp-x-ide*:
assumes *ide y*
shows (*λx. Exp x y*) = (*λx.* $Exp^{\leftarrow}$ *x y*)
  using *assms Exp-ide Exp-def comp-cod-arr cov-Exp-ide* **by** *auto*

lemma *Exp-ide-y*:
assumes *ide x*
shows (*λy. Exp x y*) = (*λy.* $Exp^{\rightarrow}$ *x y*)
  using *assms Exp-ide Exp-def comp-arr-dom cnt-Exp-ide* **by** *auto*

lemma *Uncurry-Exp-dom*:
assumes *arr f*
shows *Uncurry* (*dom f*) (*cod f*) (*Exp* (*dom f*) *f*) = *f* · *eval* (*dom f*) (*dom f*)
proof –
  have *Uncurry*[*dom f*, *cod f*] (*Exp* (*dom f*) *f*) =
        Uncurry[*dom f*, *cod f*]
         (*Curry*[*exp* (*dom f*) (*dom f*), *dom f*, *cod f*] (*f* · *eval* (*dom f*) (*dom f*)) ·

Curry[*exp* (*dom f*) (*dom f*), *dom f*, *dom f*] (*eval* (*dom f*) (*dom f*)))
unfolding *Exp-def*
using *assms Curry-Uncurry comp-arr-dom* **by** *simp*
also have ... = *Uncurry*[*dom f*, *cod f*]
(*Curry*[*exp* (*dom f*) (*dom f*), *dom f*, *cod f*]
((*f* · *eval* (*dom f*) (*dom f*)) ·
(*Curry*[*exp* (*dom f*) (*dom f*), *dom f*, *dom f*]
(*eval* (*dom f*) (*dom f*)) ⊗ *dom f*)))
using *assms comp-Curry-arr*
by (*metis comp-in-homI′ Curry-in-hom eval-in-hom*$_{ECMC}$ *ide-dom*
ide-exp in-homE)
also have ... = *f* · *eval* (*dom f*) (*dom f*)
using *assms Uncurry-Curry eval-in-hom*$_{ECMC}$ *comp-assoc* **by** *simp*
finally show *?thesis* **by** *simp*
qed

1.2.1 Exponentiation by Unity

In this section we define and develop the properties of inverse arrows *Up a* : *a* → *exp* $\mathcal{I}$ *a* and *Dn a* : *exp* $\mathcal{I}$ *a* → *a*, which exist in any closed monoidal category.

interpretation *elementary-monoidal-category C tensor unity lunit runit assoc*
using *induces-elementary-monoidal-category* **by** *blast*

abbreviation *Up*
where *Up a* ≡ *Curry*[*a*, $\mathcal{I}$, *a*] r[*a*]

abbreviation *Dn*
where *Dn a* ≡ *eval* $\mathcal{I}$ *a* · r^{-1}[*exp* $\mathcal{I}$ *a*]

lemma *isomorphic-exp-unity*:
assumes *ide a*
shows «*Up a* : *a* → *exp* $\mathcal{I}$ *a*»
and «*Dn a* : *exp* $\mathcal{I}$ *a* → *a*»
and *inverse-arrows* (*Up a*) (*Dn a*)
and *isomorphic* (*exp* $\mathcal{I}$ *a*) *a*
proof −
show *1*: «*Up a* : *a* → *exp* $\mathcal{I}$ *a*»
using *assms ide-unity Curry-in-hom* **by** *blast*
show *2*: «*Dn a* : *exp* $\mathcal{I}$ *a* → *a*»
using *assms eval-in-hom*$_{ECMC}$ [*of* $\mathcal{I}$ *a*] *runit-in-hom ide-unity* **by** *blast*
show *inverse-arrows* (*Up a*) (*Dn a*)
proof
show *ide* ((*Dn a*) · *Up a*)
by (*metis* (*no-types*, *lifting*) ‹«*Up a* : *a* → *exp* $\mathcal{I}$ *a*»›
assms comp-runit-runit′(*1*) *ide-unity in-homE comp-assoc*
runit′-naturality runit-in-hom Uncurry-Curry)
show *ide* (*Up a* · *Dn a*)
proof −

have *Up a* · *Dn a* = (*Curry*[*a*, $\mathcal{I}$, *a*] r[*a*] · *eval* $\mathcal{I}$ *a*) · r^{-1}[*exp* $\mathcal{I}$ *a*]
using *comp-assoc* **by** *simp*
also have ... =
Curry[*exp* $\mathcal{I}$ *a* ⊗ $\mathcal{I}$, $\mathcal{I}$, *a*] (r[*a*] · (*eval* $\mathcal{I}$ *a* ⊗ $\mathcal{I}$)) · r^{-1}[*exp* $\mathcal{I}$ *a*]
using *assms comp-Curry-arr*
by (*metis eval-in-hom-ax ide-unity runit-in-hom*)
also have ... =
Curry[*exp* $\mathcal{I}$ *a* ⊗ $\mathcal{I}$, $\mathcal{I}$, *a*] (*eval* $\mathcal{I}$ *a* · r[*exp* $\mathcal{I}$ *a* ⊗ $\mathcal{I}$]) · r^{-1}[*exp* $\mathcal{I}$ *a*]
using *assms runit-naturality*
by (*metis* (*no-types*, *lifting*) *eval-in-hom*$_{ECMC}$ *ide-unity in-homE*)
also have ... = (*Curry*[*exp* $\mathcal{I}$ *a*, $\mathcal{I}$, *a*] (*eval* $\mathcal{I}$ *a*) · r[*exp* $\mathcal{I}$ *a*]) · r^{-1}[*exp* $\mathcal{I}$ *a*]
by (*metis assms comp-Curry-arr eval-in-hom*$_{ECMC}$ *ide-exp ide-unity runit-commutes-with-R runit-in-hom*)
also have ... = *Curry*[*exp* $\mathcal{I}$ *a*, $\mathcal{I}$, *a*] (*eval* $\mathcal{I}$ *a*) · r[*exp* $\mathcal{I}$ *a*] · r^{-1}[*exp* $\mathcal{I}$ *a*]
using *comp-assoc* **by** *simp*
also have ... = *Curry*[*exp* $\mathcal{I}$ *a*, $\mathcal{I}$, *a*] (*eval* $\mathcal{I}$ *a*)
by (*metis assms 1 2 calculation comp-arr-ide comp-runit-runit'*(*1*) *ide-exp ide-unity seqI'*)
also have ... = *exp* $\mathcal{I}$ *a*
using *assms Curry-Uncurry*
by (*metis ide-exp ide-in-hom ide-unity Uncurry-exp*)
finally show *?thesis*
using *assms ide-exp ide-unity* **by** *presburger*
qed
qed
thus *isomorphic* (*exp* $\mathcal{I}$ *a*) *a*
by (*metis* ‹«*Up a* : *a* → *exp* $\mathcal{I}$ *a*»› *in-homE isoI isomorphicI isomorphic-symmetric*)
qed

The maps *Up* and *Dn* are natural in a.

lemma *Up-Dn-naturality*:
assumes *arr f*
shows *Exp*$^{\rightarrow}$ $\mathcal{I}$ *f* · *Up* (*dom f*) = *Up* (*cod f*) · *f*
and *Dn* (*cod f*) · *Exp*$^{\rightarrow}$ $\mathcal{I}$ *f* = *f* · *Dn* (*dom f*)
proof −
show *1*: *Exp*$^{\rightarrow}$ $\mathcal{I}$ *f* · *Up* (*dom f*) = *Up* (*cod f*) · *f*
proof −
have *Exp*$^{\rightarrow}$ $\mathcal{I}$ *f* · *Up* (*dom f*) =
Curry[*dom f*, $\mathcal{I}$, *cod f*]
((*f* · *eval* $\mathcal{I}$ (*dom f*)) · (*Curry*[*dom f*, $\mathcal{I}$, *dom f*] r[*dom f*] ⊗ $\mathcal{I}$))
using *assms comp-Curry-arr isomorphic-exp-unity*(*1*) **by** *auto*
also have ... = *Curry*[*dom f*, $\mathcal{I}$, *cod f*] (r[*cod f*] · (*f* ⊗ $\mathcal{I}$))
using *assms comp-assoc Uncurry-Curry runit-naturality* **by** *simp*
also have ... = *Up* (*cod f*) · *f*
by (*metis assms comp-Curry-arr ide-cod ide-unity in-homI runit-in-hom*)
finally show *?thesis* **by** *blast*
qed
have *Exp*$^{\rightarrow}$ $\mathcal{I}$ *f* · *inv* (*Dn* (*dom f*)) = *inv* (*Dn* (*cod f*)) · *f*

using *assms 1 isomorphic-exp-unity isomorphic-exp-unity*
by (*metis ide-cod ide-dom inverse-arrows-sym inverse-unique*)
moreover have *2*: *iso* (*Dn* (*cod f*))
using *assms isomorphic-exp-unity* [*of cod f*] **by** *auto*
moreover have *3*: *iso* (*Dn* (*dom f*))
using *assms isomorphic-exp-unity* [*of dom f*] **by** *auto*
moreover have *seq* (*inv* (*Dn* (*cod f*))) *f*
using *assms 2* **by** *auto*
ultimately show *Dn* (*cod f*) · $Exp^{\rightarrow}$ $\mathcal{I}$ *f* = *f* · *Dn* (*dom f*)
using *assms 2 3 inv-inv iso-inv-iso comp-assoc isomorphic-exp-unity*
invert-opposite-sides-of-square
[*of inv* (*eval* $\mathcal{I}$ (*cod f*) · r^{-1}[*exp* $\mathcal{I}$ (*cod f*)]) *f* $Exp^{\rightarrow}$ $\mathcal{I}$ *f*
inv (*eval* $\mathcal{I}$ (*dom f*) · r^{-1}[*exp* $\mathcal{I}$ (*dom f*)])]
by *metis*
qed

1.2.2 Internal Currying

Currying internalizes to an isomorphism between *exp* (*x* ⊗ *a*) *b* and *exp x* (*exp a b*).

abbreviation *curry*
where *curry x b c* ≡
Curry[*exp* (*x* ⊗ *b*) *c*, *x*, *exp b c*]
(*Curry*[*exp* (*x* ⊗ *b*) *c* ⊗ *x*, *b*, *c*]
(*eval* (*x* ⊗ *b*) *c* · a[*exp* (*x* ⊗ *b*) *c*, *x*, *b*]))

abbreviation *uncurry*
where *uncurry x b c* ≡
Curry[*exp x* (*exp b c*), *x* ⊗ *b*, *c*]
(*eval b c* · (*eval x* (*exp b c*) ⊗ *b*) · a^{-1}[*exp x* (*exp b c*), *x*, *b*])

lemma *internal-curry*:
assumes *ide x* **and** *ide a* **and** *ide b*
shows «*curry x a b* : *exp* (*x* ⊗ *a*) *b* → *exp x* (*exp a b*)»
and «*uncurry x a b* : *exp x* (*exp a b*) → *exp* (*x* ⊗ *a*) *b*»
and *inverse-arrows* (*curry x a b*) (*uncurry x a b*)
proof −
show *1*: «*curry x a b* : *exp* (*x* ⊗ *a*) *b* → *exp x* (*exp a b*)»
using *assms*
by (*meson assoc-in-hom comp-in-homI Curry-in-hom* $\textit{eval-in-hom}_{ECMC}$
ide-exp tensor-preserves-ide)
show *2*: «*uncurry x a b* : *exp x* (*exp a b*) → *exp* (*x* ⊗ *a*) *b*»
using *assms ide-exp* **by** *auto*
show *inverse-arrows* (*curry x a b*) (*uncurry x a b*)
(**is** *inverse-arrows*
(*Curry* (*exp* (*x* ⊗ *a*) *b*) *x* (*exp a b*)
(*Curry* (*exp* (*x* ⊗ *a*) *b* ⊗ *x*) *a b ?F*))
(*Curry* (*exp x* (*exp a b*)) (*x* ⊗ *a*) *b ?G*))
proof

have *F*: «*?F* : (*exp* (*x* ⊗ *a*) *b* ⊗ *x*) ⊗ *a* → *b*»
  using *assms ide-exp* **by** *simp*
have *G*: «*?G* : *exp x* (*exp a b*) ⊗ *x* ⊗ *a* → *b*»
  using *assms ide-exp* **by** *auto*
show *ide* (*uncurry x a b* · *curry x a b*)
proof −
  have *uncurry x a b* · *curry x a b* =
      Curry[*exp* (*x* ⊗ *a*) *b*, *x* ⊗ *a*, *b*] (*?G* · (*curry x a b* ⊗ *x* ⊗ *a*))
    using *assms F 1 ide-exp comp-Curry-arr comp-assoc* **by** *auto*
  also have ... = *Curry*[*exp* (*x* ⊗ *a*) *b*, *x* ⊗ *a*, *b*]
      (*eval a b* · (*eval x* (*exp a b*) ⊗ *a*) · a^{-1}[*exp x* (*exp a b*), *x*, *a*] ·
        (*curry x a b* ⊗ *x* ⊗ *a*))
    using *comp-assoc* **by** *simp*
  also have ... = *Curry*[*exp* (*x* ⊗ *a*) *b*, *x* ⊗ *a*, *b*]
      (*eval a b* · (*eval x* (*exp a b*) ⊗ *a*) ·
        ((*curry x a b* ⊗ *x*) ⊗ *a*) · a^{-1}[*exp* (*x* ⊗ *a*) *b*, *x*, *a*])
    using *assms 1 comp-assoc assoc′-naturality* [*of curry x a b x a*]
      ide-char in-homE
    by *metis*
  also have ... = *Curry*[*exp* (*x* ⊗ *a*) *b*, *x* ⊗ *a*, *b*]
      (*eval a b* · ((*eval x* (*exp a b*) ⊗ *a*) · ((*curry x a b* ⊗ *x*) ⊗ *a*)) ·
        a^{-1}[*exp* (*x* ⊗ *a*) *b*, *x*, *a*])
    using *comp-assoc* **by** *simp*
  also have ... = *Curry*[*exp* (*x* ⊗ *a*) *b*, *x* ⊗ *a*, *b*]
      (*eval a b* · (*Uncurry*[*x*, *exp a b*] (*curry x a b*) ⊗ *a*) ·
        a^{-1}[*exp* (*x* ⊗ *a*) *b*, *x*, *a*])
    using *assms comp-ide-self*
      interchange [*of eval x* (*exp a b*)
          Curry[*exp* (*x* ⊗ *a*) *b*, *x*, *exp a b*]
            (*Curry*[*exp* (*x* ⊗ *a*) *b* ⊗ *x*, *a*, *b*] *?F*) ⊗ *x*
          a a]
    by *fastforce*
  also have ... = *Curry*[*exp* (*x* ⊗ *a*) *b*, *x* ⊗ *a*, *b*]
      (*eval a b* ·
        (*Curry*[*exp* (*x* ⊗ *a*) *b* ⊗ *x*, *a*, *b*] *?F* ⊗ *a*) ·
          a^{-1}[*exp* (*x* ⊗ *a*) *b*, *x*, *a*])
    using *assms F ide-exp comp-assoc comp-ide-self*
      Uncurry-Curry
        [*of exp* (*x* ⊗ *a*) *b x exp a b Curry*[*exp* (*x* ⊗ *a*) *b* ⊗ *x*, *a*, *b*] *?F*]
    by *fastforce*
  also have ... = *Curry*[*exp* (*x* ⊗ *a*) *b*, *x* ⊗ *a*, *b*]
      (*eval* (*x* ⊗ *a*) *b* · a[*exp* (*x* ⊗ *a*) *b*, *x*, *a*] ·
        a^{-1}[*exp* (*x* ⊗ *a*) *b*, *x*, *a*])
    using *assms Uncurry-Curry*
    by (*metis F ide-exp comp-assoc tensor-preserves-ide*)
  also have ... = *Curry*[*exp* (*x* ⊗ *a*) *b*, *x* ⊗ *a*, *b*] (*eval* (*x* ⊗ *a*) *b*)
    using *assms Uncurry-exp* **by** *simp*
  also have ... = *exp* (*x* ⊗ *a*) *b*
    using *assms Curry-Uncurry*

by (*metis Curry-Uncurry ide-exp ide-in-hom tensor-preserves-ide Uncurry-exp*)
finally have *uncurry x a b* · *curry x a b* = *exp* (*x* ⊗ *a*) *b*
by *blast*
thus *?thesis*
using *assms* **by** *simp*
qed
show *ide* (*curry x a b* · *uncurry x a b*)
proof −
have *curry x a b* · *uncurry x a b* =
Curry[*exp x* (*exp a b*), *x*, *exp a b*]
(*Curry*[*exp* (*x* ⊗ *a*) *b* ⊗ *x*, *a*, *b*] *?F* · (*uncurry x a b* ⊗ *x*))
using *assms 2 F Curry-in-hom comp-Curry-arr* **by** *simp*
also have ... = *Curry*[*exp x* (*exp a b*), *x*, *exp a b*]
(*Curry*[*exp x* (*exp a b*) ⊗ *x*, *a*, *b*]
(*eval* (*x* ⊗ *a*) *b* · a[*exp* (*x* ⊗ *a*) *b*, *x*, *a*] ·
((*uncurry x a b* ⊗ *x*) ⊗ *a*)))
proof −
have *Curry*[*exp* (*x* ⊗ *a*) *b* ⊗ *x*, *a*, *b*] *?F* · (*uncurry x a b* ⊗ *x*) =
Curry[*exp x* (*exp a b*) ⊗ *x*, *a*, *b*] (*?F* · ((*uncurry x a b* ⊗ *x*) ⊗ *a*))
using *assms*(*1*−*2*) *2 F comp-Curry-arr ide-in-hom* **by** *auto*
thus *?thesis*
using *comp-assoc* **by** *simp*
qed
also have ... = *Curry*[*exp x* (*exp a b*), *x*, *exp a b*]
(*Curry*[*exp x* (*exp a b*) ⊗ *x*, *a*, *b*]
(*eval* (*x* ⊗ *a*) *b* ·
(*uncurry x a b* ⊗ *x* ⊗ *a*) · a[*exp x* (*exp a b*), *x*, *a*]))
using *assms 2*
assoc-naturality [*of Curry* (*exp x* (*exp a b*)) (*x* ⊗ *a*) *b ?G x a*]
by *auto*
also have ... = *Curry*[*exp x* (*exp a b*), *x*, *exp a b*]
(*Curry*[*exp x* (*exp a b*) ⊗ *x*, *a*, *b*]
(*eval a b* · (*eval x* (*exp a b*) ⊗ *a*) ·
a^{-1}[*exp x* (*exp a b*), *x*, *a*] · a[*exp x* (*exp a b*), *x*, *a*]))
using *assms Uncurry-Curry*
by (*metis G ide-exp comp-assoc tensor-preserves-ide*)
also have ... = *Curry*[*exp x* (*exp a b*), *x*, *exp a b*]
(*Curry*[*exp x* (*exp a b*) ⊗ *x*, *a*, *b*]
(*Uncurry*[*a*, *b*] (*eval x* (*exp a b*))))
using *assms*
by (*metis G arrI cod-assoc′ comp-arr-dom comp-assoc-assoc′*(*2*)
ide-exp seqE)
also have ... = *Curry*[*exp x* (*exp a b*), *x*, *exp a b*] (*eval x* (*exp a b*))
by (*simp add*: *assms*(*1*−*3*) *Curry-Uncurry* $\textit{eval-in-hom}_{ECMC}$)
also have ... = *exp x* (*exp a b*)
using *assms Curry-Uncurry Uncurry-exp*
by (*metis ide-exp ide-in-hom*)
finally have *curry x a b* · *uncurry x a b* = *exp x* (*exp a b*)

by *blast*
thus *?thesis*
using *assms* **by** *fastforce*
qed
qed
qed

Internal currying and uncurrying are the components of natural isomorphisms between the contravariant functors $Exp^{\leftarrow}$ (- $\otimes$ b) c and $Exp^{\leftarrow}$ - (exp b c).

lemma *uncurry-naturality*:
assumes *ide b* **and** *ide c* **and** *Cop.arr f*
shows *uncurry* (*Cop.cod f*) *b c* · *Exp*$^{\leftarrow}$ *f* (*exp b c*) =
Curry[*exp* (*Cop.dom f*) (*exp b c*), *Cop.cod f* ⊗ *b*, *c*]
(*eval* (*Cop.dom f* ⊗ *b*) *c* · (*uncurry* (*Cop.dom f*) *b c* ⊗ *f* ⊗ *b*))
and *Exp*$^{\leftarrow}$ (*f* ⊗ *b*) *c* · *uncurry* (*Cop.dom f*) *b c* =
Curry[*exp* (*Cop.dom f*) (*exp b c*), *Cop.cod f* ⊗ *b*, *c*]
(*eval* (*Cop.dom f* ⊗ *b*) *c* · (*uncurry* (*Cop.dom f*) *b c* ⊗ *f* ⊗ *b*))
and *uncurry* (*Cop.cod f*) *b c* · *Exp*$^{\leftarrow}$ *f* (*exp b c*) =
Exp$^{\leftarrow}$ (*f* ⊗ *b*) *c* · *uncurry* (*Cop.dom f*) *b c*
proof −
interpret *xb*: *functor Cop.comp Cop.comp* ‹λ*x*. *x* ⊗ *b*›
using *assms*(*1*) *T.fixing-ide-gives-functor-2* [*of b*]
by (*simp add*: *category-axioms dual-category.intro dual-functor.intro dual-functor.is-functor*)
interpret *F*: *functor Cop.comp C* ‹λ*x*. *Exp*$^{\leftarrow}$ *x* (*exp b c*)›
using *assms functor-cnt-Exp* **by** *blast*
have ∗: ⋀*x*. *Cop.ide x* ⟹
Uncurry (*x* ⊗ *b*) *c* (*uncurry x b c*) =
eval b c · (*eval x* (*exp b c*) ⊗ *b*) · a^{-1}[*exp x* (*exp b c*), *x*, *b*]
using *assms Uncurry-Curry Cop.ide-char* **by** *auto*
show *1*: *uncurry* (*Cop.cod f*) *b c* · *cnt-Exp f* (*exp b c*) =
Curry[*exp* (*Cop.dom f*) (*exp b c*), *Cop.cod f* ⊗ *b*, *c*]
(*eval* (*Cop.dom f* ⊗ *b*) *c* · (*uncurry* (*Cop.dom f*) *b c* ⊗ *f* ⊗ *b*))
proof −
have *uncurry* (*Cop.cod f*) *b c* · *cnt-Exp f* (*exp b c*) =
Curry[*exp* (*Cop.dom f*) (*exp b c*), *Cop.cod f* ⊗ *b*, *c*]
((*eval b c* ·
(*eval* (*Cop.cod f*) (*exp b c*) ⊗ *b*) ·
a^{-1}[*exp* (*Cop.cod f*) (*exp b c*), (*Cop.cod f*), *b*]) ·
(*cnt-Exp f* (*exp b c*) ⊗ *Cop.cod f* ⊗ *b*))
using *assms ide-exp cnt-Exp-in-hom comp-Curry-arr* **by** *auto*
also have ... = *Curry*[*exp* (*Cop.dom f*) (*exp b c*), *Cop.cod f* ⊗ *b*, *c*]
((*eval b c* ·
(*eval* (*Cop.cod f*) (*exp b c*) ⊗ *b*) ·
((*cnt-Exp f* (*exp b c*) ⊗ *Cop.cod f*) ⊗ *b*)) ·
a^{-1}[*exp* (*Cop.dom f*) (*exp b c*), *Cop.cod f*, *b*])
using *assms comp-assoc*
assoc′-naturality [*of cnt-Exp f* (*exp b c*) *Cop.cod f b*]

by *auto*
also have ... = *Curry*[*exp* (*Cop.dom f*) (*exp b c*), *Cop.cod f* ⊗ *b*, *c*]
(*Uncurry*[*b*, *c*]
(*Uncurry*[*Cop.cod f*, *exp b c*]
(*Curry*[*exp* (*Cop.dom f*) (*exp b c*), *Cop.cod f*, *exp b c*]
(*eval* (*Cop.dom f*) (*exp b c*) ·
(*exp* (*Cop.dom f*) (*exp b c*) ⊗ *f*)))) ·
a^{-1}[*exp* (*Cop.dom f*) (*exp b c*), *Cop.cod f*, *b*])
using *assms interchange* **by** *simp*
also have ... = *Curry*[*exp* (*Cop.dom f*) (*exp b c*), *Cop.cod f* ⊗ *b*, *c*]
(*eval b c* ·
((*eval* (*Cop.dom f*) (*exp b c*) ·
(*exp* (*Cop.dom f*) (*exp b c*) ⊗ *f*)) ⊗ *b*) ·
a^{-1}[*exp* (*Cop.dom f*) (*exp b c*), *Cop.cod f*, *b*])
using *assms Uncurry-Curry comp-assoc* **by** *force*
also have ... = *Curry*[*exp* (*Cop.dom f*) (*exp b c*), *Cop.cod f* ⊗ *b*, *c*]
(*eval b c* ·
((*eval* (*Cop.dom f*) (*exp b c*) ⊗ *b*) ·
((*exp* (*Cop.dom f*) (*exp b c*) ⊗ *f*) ⊗ *b*)) ·
a^{-1}[*exp* (*Cop.dom f*) (*exp b c*), *Cop.cod f*, *b*])
using *assms interchange* **by** *simp*
also have ... = *Curry*[*exp* (*Cop.dom f*) (*exp b c*), *Cop.cod f* ⊗ *b*, *c*]
((*eval b c* · (*eval* (*Cop.dom f*) (*exp b c*) ⊗ *b*) ·
a^{-1}[*exp* (*Cop.dom f*) (*exp b c*), *cod f*, *b*]) ·
(*exp* (*Cop.dom f*) (*exp b c*) ⊗ *f* ⊗ *b*))
using *assms assoc′-naturality* [*of exp* (*Cop.dom f*) (*exp b c*) *f b*] *comp-assoc*
by *simp*
also have ... = *Curry*[*exp* (*Cop.dom f*) (*exp b c*), *Cop.cod f* ⊗ *b*, *c*]
(*Uncurry*[*Cop.dom f* ⊗ *b*, *c*]
(*uncurry* (*Cop.dom f*) *b c*) ·
(*exp* (*Cop.dom f*) (*exp b c*) ⊗ *f* ⊗ *b*))
using *assms* ∗ **by** *simp*
also have ... =
Curry (*exp* (*Cop.dom f*) (*exp b c*)) (*Cop.cod f* ⊗ *b*) *c*
(*eval* (*Cop.dom f* ⊗ *b*) *c* ·
(*uncurry* (*Cop.dom f*) *b c* ⊗ (*Cop.dom f* ⊗ *b*) · (*f* ⊗ *b*)))
using *assms ide-exp internal-curry*(*2*) *interchange comp-assoc*
comp-arr-dom [*of uncurry* (*Cop.dom f*) *b c*]
by *auto*
also have ... = *Curry*[*exp* (*Cop.dom f*) (*exp b c*), *Cop.cod f* ⊗ *b*, *c*]
(*eval* (*Cop.dom f* ⊗ *b*) *c* ·
(*uncurry* (*Cop.dom f*) *b c* ⊗ *f* ⊗ *b*))
using *assms*(*1*,*3*) *comp-cod-arr interchange* **by** *fastforce*
finally show *?thesis* **by** *blast*
qed
show *2*: *Exp*$^{\leftarrow}$ (*f* ⊗ *b*) *c* · *uncurry* (*Cop.dom f*) *b c* = ...
proof −
have *Exp*$^{\leftarrow}$ (*f* ⊗ *b*) *c* · *uncurry* (*Cop.dom f*) *b c* =
Curry[*exp* (*Cop.dom f* ⊗ *b*) *c*, *Cop.cod f* ⊗ *b*, *c*]

```
          (eval (Cop.dom f ⊗ b) c · (exp (Cop.dom f ⊗ b) c ⊗ f ⊗ b)) ·
            uncurry (Cop.dom f) b c
      using assms comp-arr-dom by simp
    also have ... = Curry[exp (Cop.dom f) (exp b c), Cop.cod f ⊗ b, c]
                     ((eval (Cop.dom f ⊗ b) c ·
                        (exp (Cop.dom f ⊗ b) c ⊗ f ⊗ b)) ·
                        (uncurry (Cop.dom f) b c ⊗ Cop.cod f ⊗ b))
      using assms Curry-in-hom comp-Curry-arr by force
    also have ... = Curry[exp (Cop.dom f) (exp b c), Cop.cod f ⊗ b, c]
                     (eval (Cop.dom f ⊗ b) c ·
                       (exp (Cop.dom f ⊗ b) c · uncurry (Cop.dom f) b c
                         ⊗ (f ⊗ b) · (Cop.cod f ⊗ b)))
    proof –
      have seq (exp (Cop.dom f ⊗ b) c) (uncurry (Cop.dom f) b c)
        using assms by fastforce
      thus ?thesis
        using assms internal-curry comp-assoc interchange by simp
    qed
    also have ... = Curry[exp (Cop.dom f) (exp b c), Cop.cod f ⊗ b, c]
                     (eval (Cop.dom f ⊗ b) c ·
                       (uncurry (Cop.dom f) b c ⊗ f ⊗ b))
    proof –
      have (f ⊗ b) · (Cop.cod f ⊗ b) = f ⊗ b
        using assms interchange comp-arr-dom comp-cod-arr by simp
      thus ?thesis
        using assms internal-curry comp-cod-arr [of uncurry (Cop.dom f) b c]
        by simp
    qed
    finally show ?thesis by simp
  qed
  show uncurry (Cop.cod f) b c · Exp← f (exp b c) =
        Exp← (f ⊗ b) c · uncurry (Cop.dom f) b c
    using 1 2 by simp
qed

lemma natural-isomorphism-uncurry:
assumes ide b and ide c
shows natural-isomorphism Cop.comp C
        (λx. Exp← x (exp b c)) (λx. Exp← (x ⊗ b) c)
        (λf. uncurry (Cop.cod f) b c · Exp← f (exp b c))
proof –
  interpret xb: functor Cop.comp Cop.comp ‹λx. x ⊗ b›
    using assms(1) T.fixing-ide-gives-functor-2
    by (simp add: category-axioms dual-category.intro dual-functor.intro
        dual-functor.is-functor)
  interpret Exp-c: functor Cop.comp C ‹λx. Exp← x c›
    using assms functor-cnt-Exp by blast
  interpret F: functor Cop.comp C ‹λx. Exp← x (exp b c)›
    using assms functor-cnt-Exp by blast
```

```
  interpret G: functor Cop.comp C ‹λx. Exp← (x ⊗ b) c›
  proof –
    interpret G: composite-functor Cop.comp Cop.comp C
                 ‹λx. x ⊗ b› ‹λy. Exp← y c›
      ..
    have G.map = (λx. Exp← (x ⊗ b) c)
      by auto
    thus functor Cop.comp C (λx. Exp← (x ⊗ b) c)
      using G.functor-axioms by metis
  qed
  interpret φ: transformation-by-components Cop.comp C
               ‹λx. Exp← x (exp b c)› ‹λx. Exp← (x ⊗ b) c›
               ‹λx. uncurry x b c›
  proof
    show ⋀a. Cop.ide a ⟹
               «uncurry a b c : Exp← a (exp b c) → Exp← (a ⊗ b) c»
      using assms internal-curry(2) Cop.ide-char cnt-Exp-ide by auto
    show ⋀f. Cop.arr f ⟹
               uncurry (Cop.cod f) b c · Exp← f (exp b c) =
               Exp← (f ⊗ b) c · uncurry (Cop.dom f) b c
      using assms uncurry-naturality by simp
  qed
  have natural-isomorphism Cop.comp C
         (λx. Exp← x (exp b c)) (λx. Exp← (x ⊗ b) c) φ.map
  proof
    fix a
    assume a: Cop.ide a
    show iso (φ.map a)
      using a assms internal-curry [of a b c] φ.map-simp-ide
            inverse-arrows-sym
      by auto
  qed
  moreover have φ.map = (λf. uncurry (Cop.cod f) b c · Exp← f (exp b c))
    using assms φ.map-def by auto
  ultimately show ?thesis
    unfolding φ.map-def by simp
qed

lemma natural-isomorphism-curry:
assumes ide b and ide c
shows natural-isomorphism Cop.comp C
         (λx. Exp← (x ⊗ b) c) (λx. Exp← x (exp b c))
         (λf. curry (Cop.cod f) b c · Exp← (f ⊗ b) c)
proof –
  interpret φ: natural-isomorphism Cop.comp C
               ‹λx. Exp← x (exp b c)› ‹λx. Exp← (x ⊗ b) c›
               ‹λf. uncurry (Cop.cod f) b c · Exp← f (exp b c)›
    using assms natural-isomorphism-uncurry by blast
  interpret ψ: inverse-transformation Cop.comp C
```

‹λx. Exp← x (exp b c)› ‹λx. Exp← (x ⊗ b) c›
‹λf. uncurry (Cop.cod f) b c · Exp← f (exp b c)›
..
have *1*: ⋀*a*. *Cop.ide a* ⟹ *ψ.map a* = *curry a b c*
proof −
fix *a*
assume *a*: *Cop.ide a*
have *inverse-arrows*
(*uncurry* (*Cop.cod a*) *b c* · *Exp*← *a* (*exp b c*)) (*ψ.map a*)
using *assms a ψ.inverts-components* **by** *blast*
moreover
have *inverse-arrows*
(*uncurry* (*Cop.cod a*) *b c* · *Exp*← *a* (*exp b c*)) (*curry a b c*)
by (*metis assms a Cop.ideD*(*1*,*3*) *Cop.ide-char φ.F.preserves-ide*
φ.preserves-reflects-arr comp-arr-ide internal-curry(*3*)
inverse-arrows-sym)
ultimately show *ψ.map a* = *curry a b c*
using *internal-curry inverse-arrow-unique* **by** *simp*
qed
have *ψ.map* = (λ*f*. *curry* (*Cop.cod f*) *b c* · *Exp*← (*f* ⊗ *b*) *c*)
proof
fix *f*
show *ψ.map f* = *curry* (*Cop.cod f*) *b c* · *Exp*← (*f* ⊗ *b*) *c*
using *assms 1 ψ.inverts-components internal-curry*(*3*) *ψ.is-natural-2*
Cop.ide-char ψ.is-extensional
by *auto*
qed
thus *?thesis*
using *ψ.natural-isomorphism-axioms* **by** *simp*
qed

1.2.3 Yoneda Embedding

The internal hom provides a closed monoidal category C with a "Yoneda embedding", which is a mapping that takes each arrow g of C to a natural transformation from the contravariant functor *Exp*← - (*dom g*) to the contravariant functor *Exp*← - (*cod g*). Note that here the target category is C itself, not a category of sets and functions as in the classical case. Note also that we are talking here about ordinary functors and natural transformations. We can easily prove from general considerations that the Yoneda embedding (so-defined) is faithful. However, to obtain a fullness result requires the development of a certain amount of enriched category theory, which we do elsewhere.

lemma *yoneda-embedding*:
assumes «*g* : *a* → *b*»
shows *natural-transformation Cop.comp C*
(λ*x*. *Exp*← *x a*) (λ*x*. *Exp*← *x b*) (λ*x*. *Exp x g*)
and *Uncurry*[*a*, *b*] (*Exp a g* · *Curry*[𝓘, *a*, *a*] l[*a*]) · l^{-1}[*a*] = *g*

proof –
  interpret *Exp*: *binary-functor Cop.comp C C* ‹λ*fg*. *Exp* (*fst fg*) (*snd fg*)›
    using *functor-Exp* **by** *blast*
  interpret *Exp-g*: *natural-transformation Cop.comp C*
        ‹λ*x*. *Exp x* (*dom g*)› ‹λ*x*. *Exp x* (*cod g*)› ‹λ*x*. *Exp x g*›
    using *assms Exp.fixing-arr-gives-natural-transformation-2* [*of g*] **by** *auto*
  show *natural-transformation Cop.comp C* (λ*x*. *Exp*$^{\leftarrow}$ *x a*) (λ*x*. *Exp*$^{\leftarrow}$ *x b*)
      (λ*x*. *Exp x g*)
    using *assms Exp-x-ide Exp-x-ide Exp-g.natural-transformation-axioms*
    by *auto*
  show *Uncurry*[*a*, *b*] (*Exp a g* · *Curry*[$\mathcal{I}$, *a*, *a*] l[*a*]) · l^{-1}[*a*] = *g*
  proof –
    have *Uncurry*[*a*, *b*] (*Exp a g* · *Curry*[$\mathcal{I}$, *a*, *a*] l[*a*]) · l^{-1}[*a*] =
        (*eval a b* · (*Exp a g* ⊗ *a*) · (*Curry*[$\mathcal{I}$, *a*, *a*] l[*a*] ⊗ *a*)) · l^{-1}[*a*]
      using *assms Exp-ide lunit-in-hom*
          interchange [*of Exp a g Curry*[$\mathcal{I}$, *a*, *a*] l[*a*] *a a*]
      by *auto*
    also have ... = *g* · (*eval a a* · (*Curry*[$\mathcal{I}$, *a*, *a*] l[*a*] ⊗ *a*)) · l^{-1}[*a*]
      using *assms Uncurry-Exp-dom comp-assoc* **by** (*metis in-homE*)
    also have ... = *g* · l[*a*] · l^{-1}[*a*]
      using *assms Uncurry-Curry ide-dom ide-unity lunit-in-hom* **by** *auto*
    also have ... = *g*
      using *assms comp-arr-dom* **by** *force*
    finally show *?thesis*
      by *blast*
  qed
qed

lemma *yoneda-embedding-is-faithful*:
assumes *par g g*′ **and** (λ*x*. *Exp x g*) = (λ*x*. *Exp x g*′)
shows *g* = *g*′
proof –
  have *g* · *eval* (*dom g*) (*dom g*) = *g*′ · *eval* (*dom g*) (*dom g*)
    using *assms Uncurry-Exp-dom* **by** *metis*
  thus *g* = *g*′
    using *assms retraction-eval-ide-self retraction-is-epi*
    by (*metis epiE eval-simps*(*1*,*3*) *ide-dom seqI*)
qed

The following is a version of the key fact underlying the classical Yoneda Lemma: for any natural transformation τ from *Exp*$^{\leftarrow}$ - *a* to *Exp*$^{\leftarrow}$ - *b*, there is a fixed arrow $g : a \rightarrow b$, depending only on the single component τ *a*, such that the compositions τ *x* · *e* of an arbitrary component τ *x* with arbitrary global elements $e : \mathcal{I} \rightarrow$ *exp x a* depend on τ only via *g*, and hence only via τ *a*.

lemma *hom-transformation-expansion*:
assumes *natural-transformation*
      Cop.comp C (λ*x*. *Exp*$^{\leftarrow}$ *x a*) (λ*x*. *Exp*$^{\leftarrow}$ *x b*) τ
and *ide a* **and** *ide b*

```
shows «Uncurry[a, b] (τ a · Curry[𝓘, a, a] 𝔩[a]) · 𝔩⁻¹[a] : a → b»
and ⋀x e. ⟦ide x; «e : 𝓘 → exp x a»⟧ ⟹
          τ x · e = Exp x (Uncurry[a, b] (τ a · Curry[𝓘, a, a] 𝔩[a]) · 𝔩⁻¹[a]) · e
proof -
  interpret τ: natural-transformation Cop.comp C
              ‹λx. Exp← x a› ‹λx. Exp← x b› τ
    using assms by blast
  let ?Id-a = Curry[𝓘, a, a] 𝔩[a]
  have Id-a: «?Id-a : 𝓘 → exp a a»
    using assms ide-unity by blast
  let ?g = Uncurry[a, b] (τ a · ?Id-a) · 𝔩⁻¹[a]
  show g: «?g : a → b»
    using assms(2-3) Id-a cnt-Exp-ide by auto
  have *: ⋀x e. ⟦ide x; «e : 𝓘 → exp x a»⟧
                ⟹ τ x · e = Curry[exp x a, x, b] (?g · eval x a) · e
  proof -
    fix x e
    assume x: ide x
    assume e: «e : 𝓘 → exp x a»
    let ?e' = Uncurry x a e · 𝔩⁻¹[x]
    have e': «?e' : x → a»
      using assms(2) x e by blast
    have 1: e = Exp← ?e' a · ?Id-a
    proof -
      have Exp← ?e' a · ?Id-a =
            Curry[exp a a, x, a] (eval a a · (exp a a ⊗ ?e')) · ?Id-a
        using assms(2) e' by auto
      also have ... =
                Curry[𝓘, x, a] (eval a a · (exp a a ⊗ ?e') · (?Id-a ⊗ x))
        using assms(2) Id-a e' x comp-Curry-arr comp-assoc by auto
      also have ... = Curry[𝓘, x, a] (eval a a · (?Id-a ⊗ ?e'))
       using assms(2) e' Id-a interchange comp-arr-dom comp-cod-arr in-homE
        by (metis (no-types, lifting))
      also have ... = Curry 𝓘 x a (eval a a · (?Id-a ⊗ a) · (𝓘 ⊗ ?e'))
        using assms(2) interchange
        by (metis (no-types, lifting) e' Id-a comp-arr-ide comp-cod-arr ide-char
            ide-unity in-homE seqI)
      also have ... =
                Curry[𝓘, x, a] (Uncurry a a (Curry[𝓘, a, a] 𝔩[a]) · (𝓘 ⊗ ?e'))
        using comp-assoc by simp
      also have ... = Curry[𝓘, x, a] (𝔩[a] · (𝓘 ⊗ ?e'))
        using assms(2) Uncurry-Curry comp-assoc ide-unity lunit-in-hom
        by presburger
      also have ... = Curry[𝓘, x, a] (?e' · 𝔩[x])
        using assms(2) e' in-homE lunit-naturality
        by (metis (no-types, lifting))
      also have ... = Curry[𝓘, x, a] (Uncurry[x, a] e · 𝔩⁻¹[x] · 𝔩[x])
        using comp-assoc by simp
      also have ... = Curry[𝓘, x, a] (Uncurry[x, a] e)
```

using *assms(2) x e comp-arr-dom Uncurry-simps(2)* **by** *force*
also have ... = *e*
using *assms(2) x e Curry-Uncurry ide-unity* **by** *blast*
finally show *?thesis* **by** *simp*
qed
have τ *x* · *e* = τ *x* · *Exp*$^{\leftarrow}$ *?e′ a* · *?Id-a*
using *1* **by** *simp*
also have ... = (τ *x* · *Exp*$^{\leftarrow}$ *?e′ a*) · *?Id-a*
using *comp-assoc* **by** *simp*
also have ... = (*Exp*$^{\leftarrow}$ *?e′ b* · τ *a*) · *?Id-a*
using *e′* τ*.naturality [of ?e′]* **by** *auto*
also have ... = *Curry*[*exp a b*, *x*, *b*] (*eval a b* · (*exp a b* ⊗ *?e′*)) · τ *a* · *?Id-a*
using *assms(2) e′ comp-assoc* **by** *auto*
also have ... =
Curry[$\mathcal{I}$, *x*, *b*] ((*eval a b* · (*exp a b* ⊗ *?e′*)) · (τ *a* · *?Id-a* ⊗ *x*))
proof −
have «τ *a* · *?Id-a* : $\mathcal{I}$ → *exp a b*»
using *Id-a assms(2−3) in-homI cnt-Exp-ide*
by (*intro comp-in-homI*) *auto*
moreover have «*eval a b* · (*exp a b* ⊗ *?e′*) : *exp a b* ⊗ *x* → *b*»
using *assms(2−3) e′ ide-in-hom* **by** *blast*
ultimately show *?thesis*
using *x comp-Curry-arr* **by** *blast*
qed
also have ... = *Curry*[$\mathcal{I}$, *x*, *b*] (*eval a b* · (*exp a b* ⊗ *?e′*) · (τ *a* · *?Id-a* ⊗ *x*))
using *comp-assoc* **by** *simp*
also have ... = *Curry*[$\mathcal{I}$, *x*, *b*] (*eval a b* · (*exp a b* · τ *a* · *?Id-a* ⊗ *?e′* · *x*))
proof −
have *seq* (*exp a b*) (τ *a* · *Curry*[$\mathcal{I}$, *a*, *a*] l[*a*])
using *assms ide-exp* τ*.natural-transformation-axioms Id-a Curry-Uncurry ide-exp ide-in-hom*
by *auto*
moreover have *seq* (*Uncurry*[*x*, *a*] *e* · l^{-1}[*x*]) *x*
using *x e′* **by** *auto*
ultimately show *?thesis*
using *assms interchange* **by** *simp*
qed
also have ... = *Curry*[$\mathcal{I}$, *x*, *b*] (*eval a b* · (τ *a* · *?Id-a* ⊗ *?e′*))
proof −
have *exp a b* · τ *a* · *?Id-a* = τ *a* · *?Id-a*
using *assms(2−3) e′ ide-exp comp-ide-arr* τ*.preserves-hom cnt-Exp-ide Id-a*
by *auto*
moreover have *?e′* · *x* = *?e′*
using *e′ comp-arr-dom* **by** *blast*
ultimately show *?thesis*
using *interchange* **by** *simp*
qed
also have ... = *Curry*[$\mathcal{I}$, *x*, *b*] (*eval a b* · (τ *a* · *?Id-a* ⊗ *a*) · ($\mathcal{I}$ ⊗ *?e′*))

proof −
have $(\tau\ a \cdot ?Id\text{-}a) \cdot \mathcal{I} = \tau\ a \cdot ?Id\text{-}a$
using *assms(2) comp-arr-ide*
by (*metis Id-a comp-arr-dom in-homE comp-assoc*)
moreover have $a \cdot ?e' = ?e'$
using *e′ comp-cod-arr* **by** *blast*
moreover have *seq* $(\tau\ a \cdot Curry[\mathcal{I},\ a,\ a]\ \mathrm{l}[a])\ \mathcal{I}$
using *assms(2) cnt-Exp-ide Id-a* **by** *auto*
moreover have *seq* $a\ (Uncurry[x,\ a]\ e \cdot \mathrm{l}^{-1}[x])$
using *calculation(2) e′* **by** *auto*
ultimately show *?thesis*
using *interchange* [*of* $\tau\ a \cdot ?Id\text{-}a\ \mathcal{I}\ a\ ?e'$] **by** *simp*
qed
also have ... $= Curry[\mathcal{I},\ x,\ b]\ (eval\ a\ b \cdot (\tau\ a \cdot ?Id\text{-}a \otimes a) \cdot (\mathrm{l}^{-1}[a] \cdot \mathrm{l}[a]) \cdot$
$(\mathcal{I} \otimes eval\ x\ a \cdot (e \otimes x) \cdot \mathrm{l}^{-1}[x]))$
proof −
have $(\mathcal{I} \otimes eval\ x\ a) \cdot (\mathcal{I} \otimes (e \otimes x) \cdot \mathrm{l}^{-1}[x]) =$
$(\mathcal{I} \otimes a) \cdot (\mathcal{I} \otimes eval\ x\ a) \cdot (\mathcal{I} \otimes (e \otimes x) \cdot \mathrm{l}^{-1}[x])$
using *assms e′ L.as-nat-trans.is-natural-2 comp-lunit-lunit′(2) comp-assoc*
by (*metis* (*no-types*, *lifting*) *L.as-nat-trans.preserves-comp-2 in-homE*)
thus *?thesis*
using *assms e′ comp-assoc*
by (*elim in-homE*) *auto*
qed
also have ... $= Curry[\mathcal{I},\ x,\ b]\ (?g \cdot \mathrm{l}[a] \cdot (\mathcal{I} \otimes eval\ x\ a \cdot (e \otimes x) \cdot \mathrm{l}^{-1}[x]))$
using *comp-assoc* **by** *simp*
also have ... $= Curry[\mathcal{I},\ x,\ b]\ (?g \cdot (eval\ x\ a \cdot (e \otimes x) \cdot \mathrm{l}^{-1}[x]) \cdot \mathrm{l}[x])$
using *lunit-naturality*
by (*metis* (*no-types*, *lifting*) *e′ in-homE comp-assoc*)
also have ... $= Curry[\mathcal{I},\ x,\ b]\ (?g \cdot eval\ x\ a \cdot (e \otimes x) \cdot \mathrm{l}^{-1}[x] \cdot \mathrm{l}[x])$
using *comp-assoc* **by** *simp*
also have ... $= Curry[\mathcal{I},\ x,\ b]\ (?g \cdot eval\ x\ a \cdot (e \otimes x))$
using *x comp-arr-dom e interchange* **by** *fastforce*
also have ... $= Curry[\mathcal{I},\ x,\ b]\ ((?g \cdot eval\ x\ a) \cdot (e \otimes x))$
using *comp-assoc* **by** *simp*
also have ... $= Curry[exp\ x\ a,\ x,\ b]\ (?g \cdot eval\ x\ a) \cdot e$
using *assms(2) x e g comp-Curry-arr* **by** *auto*
finally show $\tau\ x \cdot e = Curry[exp\ x\ a,\ x,\ b]\ (?g \cdot eval\ x\ a) \cdot e$
by *blast*
qed
show $\bigwedge x\ e.\ [\![ide\ x;\ «e : \mathcal{I} \rightarrow exp\ x\ a»]\!] \Longrightarrow \tau\ x \cdot e = Exp\ x\ ?g \cdot e$
proof −
fix *x e*
assume *x*: *ide x*
assume *e*: $«e : \mathcal{I} \rightarrow exp\ x\ a»$
have $\tau\ x \cdot e = Curry[exp\ x\ a,\ x,\ b]\ (?g \cdot eval\ x\ a) \cdot e$
using *x e* $*$ *τ.natural-transformation-axioms* **by** *blast*
also have ... $= (Curry[exp\ x\ a,\ x,\ cod\ ?g]\ (?g \cdot eval\ x\ a) \cdot$
$Curry[exp\ x\ a,\ x,\ a]\ (Uncurry[x,\ a]\ (exp\ x\ a))) \cdot e$

proof –
have *Curry*[*exp x a*, *x*, *a*] (*Uncurry*[*x*, *a*] (*exp x a*)) = *exp x a*
using *assms*(*2*) *x Curry-Uncurry ide-exp ide-in-hom* **by** *force*
thus *?thesis*
using *g e comp-cod-arr comp-assoc* **by** *fastforce*
qed
also have ... = *Exp x ?g* · *e*
using *x Exp-def cod-comp g* **by** *auto*
finally show *τ x* · *e* = *Exp x ?g* · *e* **by** *blast*
qed
qed

1.3 Enriched Structure

In this section we do the main work involved in showing that a closed monoidal category is "enriched in itself". For this, we need to define, for each object *a*, an arrow *Id a* : *ℐ* → *exp a a* to serve as the "identity at *a*", and for every three objects *a*, *b*, and *c*, a "compositor" *Comp a b c* : *exp b c* ⊗ *exp a b* → *exp a c*. We also need to prove that these satisfy the appropriate unit and associativity laws. Although essentially all the work is done here, the statement and proof of the the final result is deferred to a separate theory *EnrichedCategory* so that a mutual dependence between that theory and the present one is not introduced.

interpretation *elementary-monoidal-category C tensor unity lunit runit assoc*
using *induces-elementary-monoidal-category* **by** *blast*

definition *Id*
where *Id a* ≡ *Curry*[*ℐ*, *a*, *a*] l[*a*]

lemma *Id-in-hom* [*intro*]:
assumes *ide a*
shows «*Id a* : *ℐ* → *exp a a*»
unfolding *Id-def*
using *assms Curry-in-hom lunit-in-hom* **by** *simp*

lemma *Id-simps* [*simp*]:
assumes *ide a*
shows *arr* (*Id a*)
and *dom* (*Id a*) = *ℐ*
and *cod* (*Id a*) = *exp a a*
using *assms Id-in-hom* **by** *blast+*

The next definition follows Kelly [1], section 1.6.

definition *Comp*
where *Comp a b c* ≡
Curry[*exp b c* ⊗ *exp a b*, *a*, *c*]
(*eval b c* · (*exp b c* ⊗ *eval a b*) · a[*exp b c*, *exp a b*, *a*])

lemma *Comp-in-hom* [*intro*]:
assumes *ide a* **and** *ide b* **and** *ide c*
shows «*Comp a b c* : *exp b c* ⊗ *exp a b* → *exp a c*»
  using *assms ide-exp ide-in-hom Comp-def Curry-in-hom tensor-preserves-ide*
  by *auto*

lemma *Comp-simps* [*simp*]:
assumes *ide a* **and** *ide b* **and** *ide c*
shows *arr* (*Comp a b c*)
and *dom* (*Comp a b c*) = *exp b c* ⊗ *exp a b*
and *cod* (*Comp a b c*) = *exp a c*
  using *assms Comp-in-hom in-homE* **by** *blast+*

lemma *Comp-unit-right*:
assumes *ide a* **and** *ide b* **and** *ide c*
shows «*Comp a a b* · (*exp a b* ⊗ *Id a*) : *exp a b* ⊗ $\mathcal{I}$ → *exp a b*»
and *Comp a a b* · (*exp a b* ⊗ *Id a*) = r[*exp a b*]
proof −
  show *0*: «*Comp a a b* · (*exp a b* ⊗ *Id a*) : *exp a b* ⊗ $\mathcal{I}$ → *exp a b*»
    using *assms Id-in-hom tensor-in-hom ide-in-hom ide-exp* **by** *force*
  show *Comp a a b* · (*exp a b* ⊗ *Id a*) = r[*exp a b*]
  proof (*intro runit-eqI*)
    show *1*: «*Comp a a b* · (*exp a b* ⊗ *Id a*) : *exp a b* ⊗ $\mathcal{I}$ → *exp a b*»
      by *fact*
    show *Comp a a b* · (*exp a b* ⊗ *Id a*) ⊗ $\mathcal{I}$ = (*exp a b* ⊗ ι) · a[*exp a b*, $\mathcal{I}$, $\mathcal{I}$]
    proof −
      have r[*exp a b*] · (*Comp a a b* · (*exp a b* ⊗ *Id a*) ⊗ $\mathcal{I}$) ·
              inv a[*exp a b*, $\mathcal{I}$, $\mathcal{I}$] =
            r[*exp a b*] · ((*Comp a a b* ⊗ $\mathcal{I}$) · ((*exp a b* ⊗ *Id a*) ⊗ $\mathcal{I}$)) ·
              inv a[*exp a b*, $\mathcal{I}$, $\mathcal{I}$]
        using ‹«*Comp a a b* · (*exp a b* ⊗ *Id a*) : *exp a b* ⊗ $\mathcal{I}$ → *exp a b*»› *arrI*
        by *force*
      also have ... = (r[*exp a b*] · (*Comp a a b* ⊗ $\mathcal{I}$)) ·
                        ((*exp a b* ⊗ *Id a*) ⊗ $\mathcal{I}$) · *inv* a[*exp a b*, $\mathcal{I}$, $\mathcal{I}$]
        using *comp-assoc* **by** *simp*
      also have ... = (*Comp a a b* · r[*exp a b* ⊗ *exp a a*]) ·
                        ((*exp a b* ⊗ *Id a*) ⊗ $\mathcal{I}$) · *inv* a[*exp a b*, $\mathcal{I}$, $\mathcal{I}$]
        using *assms runit-naturality*
        by (*metis Comp-simps*(*1*−*2*) *1 cod-comp in-homE*)
      also have ... = *Comp a a b* ·
                        (r[*exp a b* ⊗ *exp a a*] · ((*exp a b* ⊗ *Id a*) ⊗ $\mathcal{I}$)) ·
                          inv a[*exp a b*, $\mathcal{I}$, $\mathcal{I}$]
        using *comp-assoc* **by** *simp*
      also have ... = *Comp a a b* · ((*exp a b* ⊗ *Id a*) · r[*exp a b* ⊗ $\mathcal{I}$]) ·
                        inv a[*exp a b*, $\mathcal{I}$, $\mathcal{I}$]
        using *assms 1 runit-naturality*
        by (*metis calculation in-homE comp-assoc*)
      also have ... = *Comp a a b* · (*exp a b* ⊗ *Id a*) · r[*exp a b* ⊗ $\mathcal{I}$] ·

```
                  inv a[exp a b, 𝓘, 𝓘]
  using comp-assoc by simp
also have ... = Comp a a b · (exp a b ⊗ Id a) · (exp a b ⊗ ι)
  using assms ide-unity runit-tensor' ide-exp runit-eqI unit-in-hom-ax
    unit-triangle(1)
  by presburger
also have ... = (Curry[exp a b ⊗ exp a a, a, b]
                   (eval a b · (exp a b ⊗ eval a a) · a[exp a b, exp a a, a]) ·
                  (exp a b ⊗ Id a)) · (exp a b ⊗ ι)
  using Comp-def comp-assoc by simp
also have ... = Curry[exp a b ⊗ 𝓘, a, b]
                  ((eval a b · (exp a b ⊗ eval a a) · a[exp a b, exp a a, a]) ·
                      ((exp a b ⊗ Id a) ⊗ a)) ·
                   (exp a b ⊗ ι)
proof −
  have «exp a b ⊗ Id a : exp a b ⊗ 𝓘 → exp a b ⊗ exp a a»
    using assms by auto
  moreover have «eval a b · (exp a b ⊗ eval a a) · a[exp a b, exp a a, a] :
                   (exp a b ⊗ exp a a) ⊗ a → b»
    using assms tensor-in-hom ide-in-hom ide-exp eval-in-hom_ECMC
    by force
  ultimately show ?thesis
    using assms comp-Curry-arr by simp
qed
also have ... = r[exp a b] · (exp a b ⊗ ι)
proof −
  have 1: Uncurry[a, b]
            (Curry[exp a b ⊗ 𝓘, a, b]
               ((eval a b · (exp a b ⊗ eval a a) · a[exp a b, exp a a, a]) ·
                   ((exp a b ⊗ Id a) ⊗ a))) =
          (eval a b · (exp a b ⊗ eval a a) · a[exp a b, exp a a, a]) ·
            ((exp a b ⊗ Id a) ⊗ a)
  proof −
    have «(eval a b · (exp a b ⊗ eval a a) · a[exp a b, exp a a, a]) ·
            ((exp a b ⊗ Id a) ⊗ a) : (exp a b ⊗ 𝓘) ⊗ a → b»
      using assms tensor-in-hom ide-in-hom eval-in-hom_ECMC ide-exp
      by force
    thus ?thesis
      using assms Uncurry-Curry by auto
  qed
  also have ... = (eval a b · (exp a b ⊗ eval a a) · (exp a b ⊗ Id a ⊗ a)) ·
                    a[exp a b, 𝓘, a]
    using assms ide-exp comp-assoc assoc-naturality [of exp a b Id a a]
    by auto
  also have ... = (eval a b · (exp a b ⊗ Uncurry[a, a] (Id a))) ·
                   a[exp a b, 𝓘, a]
    using assms interchange
    by (metis (no-types, lifting) ide-exp lunit-in-hom Uncurry-Curry
        ide-unity comp-ide-self ideD(1) in-homE Id-def)
```

```
          also have ... = (eval a b · (exp a b ⊗ l[a])) · a[exp a b, I, a]
            by (metis (no-types, lifting) assms(1) lunit-in-hom Uncurry-Curry
                ide-unity Id-def)
          also have 2: ... = (eval a b · (exp a b ⊗ a) · (exp a b ⊗ l[a])) ·
                             a[exp a b, I, a]
            using assms interchange l-ide-simp by auto
         also have ... = Uncurry[a, b] (exp a b) · (exp a b ⊗ l[a]) · a[exp a b, I, a]
            using comp-assoc by simp
          also have ... = Uncurry a b r[exp a b]
            using assms triangle ide-exp 2 comp-assoc by auto
          finally have Uncurry[a, b]
                         (Curry[exp a b ⊗ I, a, b]
                            ((eval a b · (exp a b ⊗ eval a a) ·
                                a[exp a b, exp a a, a]) ·
                                 ((exp a b ⊗ Id a) ⊗ a))) =
                        Uncurry[a, b] r[exp a b]
            by blast
          hence Curry[exp a b ⊗ I, a, b]
                   ((eval a b · (exp a b ⊗ eval a a) · a[exp a b, exp a a, a]) ·
                       ((exp a b ⊗ Id a) ⊗ a)) =
                 r[exp a b]
            using assms 1 Curry-Uncurry runit-in-hom by force
          thus ?thesis
            by presburger
        qed
        finally have r[exp a b] ·
                       (Comp a a b · (exp a b ⊗ Id a) ⊗ I) · inv a[exp a b, I, I] =
                     r[exp a b] · (exp a b ⊗ ι)
          by blast
        hence (Comp a a b · (exp a b ⊗ Id a) ⊗ I) · inv a[exp a b, I, I] =
                exp a b ⊗ ι
          using assms ide-exp iso-cancel-left [of r[exp a b]] iso-runit by fastforce
        thus ?thesis
         by (metis assms(1−2) 0 R.as-nat-trans.is-natural-1 comp-assoc-assoc′(2)
              ide-exp ide-unity in-homE comp-assoc)
      qed
    qed
  qed

  lemma Comp-unit-left:
  assumes ide a and ide b and ide c
  shows «Comp a b b · (Id b ⊗ exp a b) : I ⊗ exp a b → exp a b»
  and Comp a b b · (Id b ⊗ exp a b) = l[exp a b]
  proof −
    show 0: «Comp a b b · (Id b ⊗ exp a b) : I ⊗ exp a b → exp a b»
      using assms ide-exp by simp
    show Comp a b b · (Id b ⊗ exp a b) = l[exp a b]
    proof (intro lunit-eqI)
      show «Comp a b b · (Id b ⊗ exp a b) : I ⊗ exp a b → exp a b»
```

by *fact*
show $\mathcal{I} \otimes$ *Comp a b b* $\cdot$ (*Id b* $\otimes$ *exp a b*) = ($\iota \otimes$ *exp a b*) $\cdot$ $\mathrm{a}^{-1}[\mathcal{I}, \mathcal{I},$ *exp a b*]
proof −
have l[*exp a b*] $\cdot$ ($\mathcal{I} \otimes$ *Comp a b b* $\cdot$ (*Id b* $\otimes$ *exp a b*)) $\cdot$ a[$\mathcal{I}$, $\mathcal{I}$, *exp a b*] =
l[*exp a b*] $\cdot$ (($\mathcal{I} \otimes$ *Comp a b b*) $\cdot$ ($\mathcal{I} \otimes$ *Id b* $\otimes$ *exp a b*)) $\cdot$ a[$\mathcal{I}$, $\mathcal{I}$, *exp a b*]
using *assms 0 interchange* [*of* $\mathcal{I}$ $\mathcal{I}$ *Comp a b b Id b* $\otimes$ *exp a b*] **by** *auto*
also have ... = (l[*exp a b*] $\cdot$ ($\mathcal{I} \otimes$ *Comp a b b*)) $\cdot$
($\mathcal{I} \otimes$ *Id b* $\otimes$ *exp a b*) $\cdot$ a[$\mathcal{I}$, $\mathcal{I}$, *exp a b*]
using *comp-assoc* **by** *simp*
also have ... = (*Comp a b b* $\cdot$ l[*exp b b* $\otimes$ *exp a b*]) $\cdot$ ($\mathcal{I} \otimes$ *Id b* $\otimes$ *exp a b*) $\cdot$
a[$\mathcal{I}$, $\mathcal{I}$, *exp a b*]
using *assms lunit-naturality*
by (*metis 0 Comp-simps*(*1*−*2*) *cod-comp in-homE*)
also have ... = *Comp a b b* $\cdot$
(l[*exp b b* $\otimes$ *exp a b*] $\cdot$ ($\mathcal{I} \otimes$ *Id b* $\otimes$ *exp a b*)) $\cdot$
a[$\mathcal{I}$, $\mathcal{I}$, *exp a b*]
using *comp-assoc* **by** *simp*
also have ... =
(*Comp a b b* $\cdot$ (*Id b* $\otimes$ *exp a b*)) $\cdot$ l[$\mathcal{I} \otimes$ *exp a b*] $\cdot$ a[$\mathcal{I}$, $\mathcal{I}$, *exp a b*]
using *assms 0 lunit-naturality calculation in-homE comp-assoc* **by** *metis*
also have ... = (*Comp a b b* $\cdot$ (*Id b* $\otimes$ *exp a b*)) $\cdot$ ($\iota \otimes$ *exp a b*)
using *assms*(*1*−*2*) *ide-exp ide-unity lunit-eqI lunit-tensor′ unit-in-hom-ax*
unit-triangle(*2*)
by *presburger*
also have ... = l[*exp a b*] $\cdot$ ($\iota \otimes$ *exp a b*)
proof (*unfold Comp-def*)
have (*Curry*[*exp b b* $\otimes$ *exp a b*, *a*, *b*]
(*eval b b* $\cdot$ (*exp b b* $\otimes$ *eval a b*) $\cdot$ a[*exp b b*, *exp a b*, *a*]) $\cdot$
(*Id b* $\otimes$ *exp a b*)) $\cdot$
($\iota \otimes$ *exp a b*) =
Curry[$\mathcal{I} \otimes$ *exp a b*, *a*, *b*]
((*eval b b* $\cdot$ (*exp b b* $\otimes$ *eval a b*) $\cdot$ a[*exp b b*, *exp a b*, *a*]) $\cdot$
((*Id b* $\otimes$ *exp a b*) $\otimes$ *a*)) $\cdot$
($\iota \otimes$ *exp a b*)
proof −
have «*eval b b* $\cdot$ (*exp b b* $\otimes$ *eval a b*) $\cdot$ a[*exp b b*, *exp a b*, *a*]
: (*exp b b* $\otimes$ *exp a b*) $\otimes$ *a* $\rightarrow$ *b*»
using *assms ide-exp tensor-in-hom ide-in-hom ide-exp eval-in-hom*$_{ECMC}$
by *force*
moreover have «*Id b* $\otimes$ *exp a b* : $\mathcal{I} \otimes$ *exp a b* $\rightarrow$ *exp b b* $\otimes$ *exp a b*»
using *assms ide-exp* **by** *force*
ultimately show *?thesis*
using *assms comp-Curry-arr* **by** *force*
qed
also have ... = *Curry*[$\mathcal{I} \otimes$ *exp a b*, *a*, *b*]
(*eval b b* $\cdot$ ((*exp b b* $\otimes$ *eval a b*) $\cdot$ (*Id b* $\otimes$ *exp a b* $\otimes$ *a*)) $\cdot$
a[$\mathcal{I}$, *exp a b*, *a*]) $\cdot$
($\iota \otimes$ *exp a b*)
using *assms assoc-naturality* [*of Id b exp a b a*] *ide-exp comp-assoc*

by *force*
also have ... = $Curry[\mathcal{I} \otimes exp\ a\ b,\ a,\ b]$
$((eval\ b\ b \cdot (Id\ b \otimes Uncurry\ a\ b\ (exp\ a\ b))) \cdot$
$a[\mathcal{I},\ exp\ a\ b,\ a]) \cdot$
$(\iota \otimes exp\ a\ b)$
by (*simp add*: *assms Uncurry-exp comp-cod-arr comp-assoc interchange*)
also have ... = $Curry[\mathcal{I} \otimes exp\ a\ b,\ a,\ b]$
$((eval\ b\ b \cdot (Id\ b \otimes eval\ a\ b)) \cdot a[\mathcal{I},\ exp\ a\ b,\ a]) \cdot$
$(\iota \otimes exp\ a\ b)$
using *assms comp-arr-dom*
by (*metis eval-in-hom*$_{ECMC}$ *in-homE*)
also have ... = $Curry[\mathcal{I} \otimes exp\ a\ b,\ a,\ b]$
$(((eval\ b\ b \cdot (Id\ b \otimes b)) \cdot (\mathcal{I} \otimes eval\ a\ b)) \cdot a[\mathcal{I},\ exp\ a\ b,\ a]) \cdot$
$(\iota \otimes exp\ a\ b)$
proof −
have $Id\ b \otimes eval\ a\ b = (Id\ b \otimes b) \cdot (\mathcal{I} \otimes eval\ a\ b)$
using *assms interchange*
by (*metis Id-simps*(*1*−*2*) *comp-arr-dom comp-ide-arr eval-in-hom*$_{ECMC}$
ide-in-hom seqI')
thus *?thesis* **using** *comp-assoc* **by** *simp*
qed
also have ... = $Curry[\mathcal{I} \otimes exp\ a\ b,\ a,\ b]$
$((l[b] \cdot (\mathcal{I} \otimes eval\ a\ b)) \cdot a[\mathcal{I},\ exp\ a\ b,\ a]) \cdot$
$(\iota \otimes exp\ a\ b)$
using *assms Id-def Uncurry-Curry lunit-in-hom ide-unity* **by** *simp*
also have ... = $Curry[\mathcal{I} \otimes exp\ a\ b,\ a,\ b]$
$(eval\ a\ b \cdot l[exp\ a\ b \otimes a] \cdot a[\mathcal{I},\ exp\ a\ b,\ a]) \cdot$
$(\iota \otimes exp\ a\ b)$
using *assms lunit-naturality eval-in-hom*$_{ECMC}$ *in-homE lunit-naturality*
comp-assoc
by *metis*
also have ... = $Curry[\mathcal{I} \otimes exp\ a\ b,\ a,\ b]$
$(Uncurry[a,\ b]\ l[exp\ a\ b]) \cdot (\iota \otimes exp\ a\ b)$
using *assms ide-exp lunit-tensor'* **by** *force*
also have ... = $l[exp\ a\ b] \cdot (\iota \otimes exp\ a\ b)$
using *assms Curry-Uncurry lunit-in-hom ide-exp* **by** *auto*
finally show $(Curry[exp\ b\ b \otimes exp\ a\ b,\ a,\ b]$
$(eval\ b\ b \cdot (exp\ b\ b \otimes eval\ a\ b) \cdot a[exp\ b\ b,\ exp\ a\ b,\ a]) \cdot$
$(Id\ b \otimes exp\ a\ b)) \cdot$
$(\iota \otimes exp\ a\ b) =$
$l[exp\ a\ b] \cdot (\iota \otimes exp\ a\ b)$
by *blast*
qed
finally have *1*: $l[exp\ a\ b] \cdot$
$(\mathcal{I} \otimes Comp\ a\ b\ b \cdot (Id\ b \otimes exp\ a\ b)) \cdot a[\mathcal{I},\ \mathcal{I},\ exp\ a\ b] =$
$l[exp\ a\ b] \cdot (\iota \otimes exp\ a\ b)$
by *blast*
have $(\mathcal{I} \otimes Comp\ a\ b\ b \cdot (Id\ b \otimes exp\ a\ b)) \cdot a[\mathcal{I},\ \mathcal{I},\ exp\ a\ b] =$
$(inv\ l[exp\ a\ b] \cdot l[exp\ a\ b]) \cdot$

$(\mathcal{I} \otimes \textit{Comp a b b} \cdot (\textit{Id b} \otimes \textit{exp a b})) \cdot \mathrm{a}[\mathcal{I}, \mathcal{I}, \textit{exp a b}]$
using *assms comp-cod-arr* **by** *simp*
also have ... $= (\textit{inv } \mathrm{l}[\textit{exp a b}] \cdot \mathrm{l}[\textit{exp a b}]) \cdot (\iota \otimes \textit{exp a b})$
using *1 comp-assoc* **by** *simp*
also have ... $= \iota \otimes \textit{exp a b}$
using *assms comp-cod-arr* [*of* $\iota \otimes \textit{exp a b } \mathrm{l}^{-1}[\textit{exp a b}] \cdot \mathrm{l}[\textit{exp a b}]$] *arrI*
by *auto*
finally have $(\mathcal{I} \otimes \textit{Comp a b b} \cdot (\textit{Id b} \otimes \textit{exp a b})) \cdot \mathrm{a}[\mathcal{I}, \mathcal{I}, \textit{exp a b}] =$
$\iota \otimes \textit{exp a b}$
by *blast*
thus *?thesis*
using *assms(1−2) 0 L.as-nat-trans.is-natural-1 comp-assoc-assoc′(1)*
ide-exp ide-unity in-homE comp-assoc
by *metis*
qed
qed
qed

lemma *Comp-assoc*$_{ECMC}$:
assumes *ide a* **and** *ide b* **and** *ide c* **and** *ide d*
shows «$\textit{Comp a b d} \cdot (\textit{Comp b c d} \otimes \textit{exp a b})$:
$(\textit{exp c d} \otimes \textit{exp b c}) \otimes \textit{exp a b} \rightarrow \textit{exp a d}$»
and $\textit{Comp a b d} \cdot (\textit{Comp b c d} \otimes \textit{exp a b}) =$
$\textit{Comp a c d} \cdot (\textit{exp c d} \otimes \textit{Comp a b c}) \cdot \mathrm{a}[\textit{exp c d}, \textit{exp b c}, \textit{exp a b}]$
proof −
show «$\textit{Comp a b d} \cdot (\textit{Comp b c d} \otimes \textit{exp a b})$:
$(\textit{exp c d} \otimes \textit{exp b c}) \otimes \textit{exp a b} \rightarrow \textit{exp a d}$»
using *assms* **by** *auto*
show $\textit{Comp a b d} \cdot (\textit{Comp b c d} \otimes \textit{exp a b}) =$
$\textit{Comp a c d} \cdot (\textit{exp c d} \otimes \textit{Comp a b c}) \cdot \mathrm{a}[\textit{exp c d}, \textit{exp b c}, \textit{exp a b}]$
proof −
have *1*: $\textit{Uncurry}[a, d]\ (\textit{Comp a c d} \cdot (\textit{exp c d} \otimes \textit{Comp a b c}) \cdot$
$\mathrm{a}[\textit{exp c d}, \textit{exp b c}, \textit{exp a b}]) =$
$\textit{Uncurry}[a, d]\ (\textit{Comp a b d} \cdot (\textit{Comp b c d} \otimes \textit{exp a b}))$
proof −
have $\textit{Uncurry}[a, d]$
$(\textit{Comp a c d} \cdot (\textit{exp c d} \otimes \textit{Comp a b c}) \cdot$
$\mathrm{a}[\textit{exp c d}, \textit{exp b c}, \textit{exp a b}]) =$
$\textit{Uncurry}[a, d]$
$(\textit{Curry}[\textit{exp c d} \otimes \textit{exp a c}, a, d]$
$(\textit{eval c d} \cdot (\textit{exp c d} \otimes \textit{eval a c}) \cdot \mathrm{a}[\textit{exp c d}, \textit{exp a c}, a]) \cdot$
$(\textit{exp c d} \otimes \textit{Comp a b c}) \cdot \mathrm{a}[\textit{exp c d}, \textit{exp b c}, \textit{exp a b}])$
using *Comp-def* **by** *simp*
also have ... $= \textit{Uncurry}[a, d]$
$(\textit{Curry}[(\textit{exp c d} \otimes \textit{exp b c}) \otimes \textit{exp a b}, a, d]$
$((\textit{eval c d} \cdot (\textit{exp c d} \otimes \textit{eval a c}) \cdot \mathrm{a}[\textit{exp c d}, \textit{exp a c}, a]) \cdot$
$((\textit{exp c d} \otimes \textit{Comp a b c}) \cdot$
$\mathrm{a}[\textit{exp c d}, \textit{exp b c}, \textit{exp a b}] \otimes a)))$
using *assms*

comp-Curry-arr
[*of a* (*exp c d* ⊗ *Comp a b c*) · a[*exp c d*, *exp b c*, *exp a b*]
(*exp c d* ⊗ *exp b c*) ⊗ *exp a b exp c d* ⊗ *exp a c*
eval c d · (*exp c d* ⊗ *eval a c*) · a[*exp c d*, *exp a c*, *a*] *d*]
by *auto*
also have ... = *eval c d* ·(*exp c d* ⊗ *eval a c*) ·
(a[*exp c d*, *exp a c*, *a*] · ((*exp c d* ⊗ *Comp a b c*) ⊗ *a*)) ·
(a[*exp c d*, *exp b c*, *exp a b*] ⊗ *a*)
using *assms Uncurry-Curry ide-exp interchange comp-assoc* **by** *simp*
also have ... = *eval c d* ·
(*exp c d* ⊗ *eval a c*) ·
(a[*exp c d*, *exp a c*, *a*] ·
((*exp c d* ⊗
Curry[*exp b c* ⊗ *exp a b*, *a*, *c*]
(*eval b c* · (*exp b c* ⊗ *eval a b*) · a[*exp b c*, *exp a b*, *a*]))
⊗ *a*)) ·
(a[*exp c d*, *exp b c*, *exp a b*] ⊗ *a*)
unfolding *Comp-def* **by** *simp*
also have ... = *eval c d* ·
(*exp c d* ⊗ *eval a c*) ·
((*exp c d* ⊗
Curry[*exp b c* ⊗ *exp a b*, *a*, *c*]
(*eval b c* · (*exp b c* ⊗ *eval a b*) · a[*exp b c*, *exp a b*, *a*])
⊗ *a*) ·
a[*exp c d*, *exp b c* ⊗ *exp a b*, *a*]) ·
(a[*exp c d*, *exp b c*, *exp a b*] ⊗ *a*)
using *assms assoc-naturality* [*of exp c d - a*] *Comp-def Comp-simps*(*1*−*3*)
ide-exp ide-char
by (*metis* (*no-types*, *lifting*) *mem-Collect-eq*)
also have ... = *eval c d* ·
((*exp c d* ⊗ *eval a c*) ·
(*exp c d* ⊗
Curry[*exp b c* ⊗ *exp a b*, *a*, *c*]
(*eval b c* · (*exp b c* ⊗ *eval a b*) · a[*exp b c*, *exp a b*, *a*])
⊗ *a*)) ·
a[*exp c d*, *exp b c* ⊗ *exp a b*, *a*] ·
(a[*exp c d*, *exp b c*, *exp a b*] ⊗ *a*)
using *comp-assoc* **by** *simp*
also have ... = *eval c d* ·
(*exp c d* ⊗
Uncurry[*a*, *c*]
(*Curry*[*exp b c* ⊗ *exp a b*, *a*, *c*]
(*eval b c* · (*exp b c* ⊗ *eval a b*) ·
a[*exp b c*, *exp a b*, *a*]))) ·
a[*exp c d*, *exp b c* ⊗ *exp a b*, *a*] ·
(a[*exp c d*, *exp b c*, *exp a b*] ⊗ *a*)
using *assms Comp-def Comp-in-hom interchange* **by** *auto*
also have ... = *eval c d* ·
(*exp c d* ⊗ (*eval b c* · (*exp b c* ⊗ *eval a b*) ·

a[*exp b c*, *exp a b*, *a*])) ·
a[*exp c d*, *exp b c* ⊗ *exp a b*, *a*] ·
(a[*exp c d*, *exp b c*, *exp a b*] ⊗ *a*)
using *assms ide-exp tensor-in-hom ide-in-hom ide-exp eval-in-hom*$_{ECMC}$
assoc-in-hom Uncurry-Curry
by *force*
also have ... = *eval c d* ·
((*exp c d* ⊗ *eval b c*) ·
(*exp c d* ⊗ *exp b c* ⊗ *eval a b*) ·
(*exp c d* ⊗ a[*exp b c*, *exp a b*, *a*])) ·
a[*exp c d*, *exp b c* ⊗ *exp a b*, *a*] ·
(a[*exp c d*, *exp b c*, *exp a b*] ⊗ *a*)
using *assms ide-exp tensor-in-hom interchange* **by** *auto*
also have ... = *eval c d* ·
(*exp c d* ⊗ *eval b c*) ·
(*exp c d* ⊗ *exp b c* ⊗ *eval a b*) ·
(*exp c d* ⊗ a[*exp b c*, *exp a b*, *a*]) ·
a[*exp c d*, *exp b c* ⊗ *exp a b*, *a*] ·
(a[*exp c d*, *exp b c*, *exp a b*] ⊗ *a*)
using *comp-assoc* **by** *simp*
finally have ∗: *Uncurry*[*a*, *d*] (*Comp a c d* · (*exp c d* ⊗ *Comp a b c*) ·
a[*exp c d*, *exp b c*, *exp a b*]) =
eval c d ·
(*exp c d* ⊗ *eval b c*) ·
(*exp c d* ⊗ *exp b c* ⊗ *eval a b*) ·
(*exp c d* ⊗ a[*exp b c*, *exp a b*, *a*]) ·
a[*exp c d*, *exp b c* ⊗ *exp a b*, *a*] ·
(a[*exp c d*, *exp b c*, *exp a b*] ⊗ *a*)
by *blast*
have *Uncurry*[*a*, *d*] (*Comp a b d* · (*Comp b c d* ⊗ *exp a b*)) =
Uncurry[*a*, *d*]
(*Curry*[*exp b d* ⊗ *exp a b*, *a*, *d*]
(*eval b d* · (*exp b d* ⊗ *eval a b*) · a[*exp b d*, *exp a b*, *a*]) ·
(*Comp b c d* ⊗ *exp a b*))
using *Comp-def* **by** *simp*
also have ... = *Uncurry*[*a*, *d*]
(*Curry*[(*exp c d* ⊗ *exp b c*) ⊗ *exp a b*, *a*, *d*]
(*eval b d* · (*exp b d* ⊗ *eval a b*) · a[*exp b d*, *exp a b*, *a*] ·
((*Comp b c d* ⊗ *exp a b*) ⊗ *a*)))
proof −
have «*Comp b c d* ⊗ *exp a b* :
(*exp c d* ⊗ *exp b c*) ⊗ *exp a b* → *exp b d* ⊗ *exp a b*»
using *assms ide-exp* **by** *force*
moreover have «*eval b d* · (*exp b d* ⊗ *eval a b*) · a[*exp b d*, *exp a b*, *a*]
: (*exp b d* ⊗ *exp a b*) ⊗ *a* → *d*»
using *assms ide-exp tensor-in-hom ide-in-hom ide-exp eval-in-hom*$_{ECMC}$
by *force*
ultimately show *?thesis*
using *comp-Curry-arr assms comp-assoc* **by** *auto*

qed
also have ... = *eval b d* · (*exp b d* ⊗ *eval a b*) · a[*exp b d*, *exp a b*, *a*] ·
((*Comp b c d* ⊗ *exp a b*) ⊗ *a*)
using *assms ide-exp Uncurry-Curry* **by** *force*
also have ... = *eval b d* ·
((*exp b d* ⊗ *eval a b*) · (*Comp b c d* ⊗ *exp a b* ⊗ *a*)) ·
a[*exp c d* ⊗ *exp b c*, *exp a b*, *a*]
using *assms ide-exp comp-assoc*
assoc-naturality [*of Comp b c d exp a b a*]
by *force*
also have ... = *eval b d* · (*Comp b c d* ⊗ *eval a b*) ·
a[*exp c d* ⊗ *exp b c*, *exp a b*, *a*]
by (*simp add*: *assms comp-arr-dom comp-cod-arr interchange*)
also have ... = *eval b d* ·
(*Curry*[*exp c d* ⊗ *exp b c*, *b*, *d*]
(*eval c d* · (*exp c d* ⊗ *eval b c*) · a[*exp c d*, *exp b c*, *b*])
⊗ *eval a b*) ·
a[*exp c d* ⊗ *exp b c*, *exp a b*, *a*]
unfolding *Comp-def* **by** *simp*
also have ... = *eval b d* ·
((*Curry*[*exp c d* ⊗ *exp b c*, *b*, *d*]
(*eval c d* · (*exp c d* ⊗ *eval b c*) · a[*exp c d*, *exp b c*, *b*])
⊗ *b*) ·
((*exp c d* ⊗ *exp b c*) ⊗ *eval a b*)) ·
a[*exp c d* ⊗ *exp b c*, *exp a b*, *a*]
by (*metis* (*no-types*, *lifting*) *assms Comp-def Comp-simps*(*1*−*2*)
comp-arr-dom comp-cod-arr eval-simps(*1*,*3*) *interchange*)
also have ... = *Uncurry*[*b*, *d*]
(*Curry*[*exp c d* ⊗ *exp b c*, *b*, *d*]
(*eval c d* · (*exp c d* ⊗ *eval b c*) · a[*exp c d*, *exp b c*, *b*])) ·
((*exp c d* ⊗ *exp b c*) ⊗ *eval a b*) ·
a[*exp c d* ⊗ *exp b c*, *exp a b*, *a*]
using *comp-assoc* **by** *simp*
also have ... = *eval c d* ·
(*exp c d* ⊗ *eval b c*) ·
(a[*exp c d*, *exp b c*, *b*] ·
((*exp c d* ⊗ *exp b c*) ⊗ *eval a b*)) ·
a[*exp c d* ⊗ *exp b c*, *exp a b*, *a*]
using *assms ide-exp Uncurry-Curry comp-assoc* **by** *auto*
also have ... = *eval c d* ·
(*exp c d* ⊗ *eval b c*) ·
((*exp c d* ⊗ *exp b c* ⊗ *eval a b*) ·
a[*exp c d*, *exp b c*, *exp a b* ⊗ *a*]) ·
a[*exp c d* ⊗ *exp b c*, *exp a b*, *a*]
using *assoc-naturality* [*of exp c d exp b c eval a b*]
by (*metis assms arr-cod cod-cod Curry-in-hom dom-dom eval-in-hom*$_{ECMC}$
ide-exp in-homE)
also have ... = *eval c d* ·
(*exp c d* ⊗ *eval b c*) ·

```
                    (exp c d ⊗ exp b c ⊗ eval a b) ·
                      a[exp c d, exp b c, exp a b ⊗ a] ·
                        a[exp c d ⊗ exp b c, exp a b, a]
      using comp-assoc by simp
    finally have **: Uncurry[a, d] (Comp a b d · (Comp b c d ⊗ exp a b)) =
                  eval c d ·
                    (exp c d ⊗ eval b c) ·
                      (exp c d ⊗ exp b c ⊗ eval a b) ·
                        a[exp c d, exp b c, exp a b ⊗ a] ·
                          a[exp c d ⊗ exp b c, exp a b, a]
      by blast
    show ?thesis
      using * ** assms ide-exp pentagon by force
  qed
  have Comp a b d · (Comp b c d ⊗ exp a b) =
        Curry[(exp c d ⊗ exp b c) ⊗ exp a b, a, d]
          (Uncurry[a, d] (Comp a b d · (Comp b c d ⊗ exp a b)))
    using assms ide-exp Curry-Uncurry by fastforce
  also have ... = Curry[(exp c d ⊗ exp b c) ⊗ exp a b, a, d]
                  (Uncurry[a, d] (Comp a c d · (exp c d ⊗ Comp a b c) ·
                                  a[exp c d, exp b c, exp a b]))
    using 1 by simp
  also have ... = Comp a c d · (exp c d ⊗ Comp a b c) ·
                  a[exp c d, exp b c, exp a b]
    using assms ide-exp Curry-Uncurry by simp
  finally show Comp a b d · (Comp b c d ⊗ exp a b) =
              Comp a c d · (exp c d ⊗ Comp a b c) · a[exp c d, exp b c, exp a b]
    by blast
qed
qed

end

end
```

1.4 Cartesian Closed Monoidal Categories

A *cartesian closed monoidal category* is a cartesian monoidal category that is a closed monoidal category with respect to a chosen product. This is not quite the same thing as a cartesian closed category, because a cartesian monoidal category (being a monoidal category) has chosen structure (the tensor, associators, and unitors), whereas we have defined a cartesian closed category to be an abstract category satisfying certain properties that are expressed without assuming any chosen structure.

```
theory CartesianClosedMonoidalCategory
imports Category3.CartesianClosedCategory MonoidalCategory.CartesianMonoidalCategory
    ClosedMonoidalCategory
```

begin

locale *cartesian-closed-monoidal-category* =
cartesian-monoidal-category +
closed-monoidal-category

locale *elementary-cartesian-closed-monoidal-category* =
cartesian-monoidal-category +
elementary-closed-monoidal-category
begin

lemmas *prod-eq-tensor* [*simp*]

end

The following is the main purpose for the current theory: to show that a cartesian closed category with chosen structure determines a cartesian closed monoidal category.

context *elementary-cartesian-closed-category*
begin

interpretation *CMC*: *cartesian-monoidal-category C Prod* α ι
using *extends-to-cartesian-monoidal-category*$_{ECC}$ **by** *blast*

interpretation *CMC*: *closed-monoidal-category C Prod* α ι
using *CMC.T.is-extensional interchange left-adjoint-prod*
by *unfold-locales*
(*auto simp add*: *left-adjoint-functor.ex-terminal-arrow*)

lemma *extends-to-closed-monoidal-category*$_{ECCC}$:
shows *closed-monoidal-category C Prod* α ι
..

lemma *extends-to-cartesian-closed-monoidal-category*$_{ECCC}$:
shows *cartesian-closed-monoidal-category C Prod* α ι
..

interpretation *CMC*: *elementary-monoidal-category*
C CMC.tensor CMC.unity CMC.lunit CMC.runit CMC.assoc
using *CMC.induces-elementary-monoidal-category* **by** *blast*

interpretation *CMC*: *elementary-closed-monoidal-category*
C Prod α ι *exp eval curry*
using *eval-in-hom-ax curry-in-hom uncurry-curry-ax curry-uncurry-ax*
by *unfold-locales auto*

lemma *extends-to-elementary-closed-monoidal-category*$_{ECCC}$:
shows *elementary-closed-monoidal-category C Prod* α ι *exp eval curry*
..

lemma *extends-to-elementary-cartesian-closed-monoidal-category*$_{ECCC}$:
shows *elementary-cartesian-closed-monoidal-category C Prod α ι exp eval curry*
..

end

context *elementary-cartesian-closed-monoidal-category*
begin

interpretation *elementary-monoidal-category C tensor unity lunit runit assoc*
using *induces-elementary-monoidal-category* **by** *blast*

The following fact is not used in the present article, but it is a natural and likely useful lemma for which I constructed a proof at one point. The proof requires cartesianness; I suspect this is essential, but I am not absolutely certain of it.

lemma *isomorphic-exp-prod*:
assumes *ide a* **and** *ide b* **and** *ide x*
shows «⟨*Curry*[*exp x* (*a* ⊗ *b*), *x*, *a*] ($\mathfrak{p}_1$[*a*, *b*] · *eval x* (*a* ⊗ *b*)),
Curry[*exp x* (*a* ⊗ *b*), *x*, *b*] ($\mathfrak{p}_0$[*a*, *b*] · *eval x* (*a* ⊗ *b*))⟩
: *exp x* (*a* ⊗ *b*) → *exp x a* ⊗ *exp x b*»
(**is** «⟨*?C*, *?D*⟩ : *exp x* (*a* ⊗ *b*) → *exp x a* ⊗ *exp x b*»)
and «*Curry*[*exp x a* ⊗ *exp x b*, *x*, *a* ⊗ *b*]
⟨*eval x a* · ⟨$\mathfrak{p}_1$[*exp x a*, *exp x b*] · $\mathfrak{p}_1$[*exp x a* ⊗ *exp x b*, *x*],
$\mathfrak{p}_0$[*exp x a* ⊗ *exp x b*, *x*]⟩,
eval x b · ⟨$\mathfrak{p}_0$[*exp x a*, *exp x b*] · $\mathfrak{p}_1$[*exp x a* ⊗ *exp x b*, *x*],
$\mathfrak{p}_0$[*exp x a* ⊗ *exp x b*, *x*]⟩⟩
: *exp x a* ⊗ *exp x b* → *exp x* (*a* ⊗ *b*)»
(**is** «*Curry*[*exp x a* ⊗ *exp x b*, *x*, *a* ⊗ *b*] ⟨*?A*, *?B*⟩
: *exp x a* ⊗ *exp x b* → *exp x* (*a* ⊗ *b*)»)
and *inverse-arrows*
(*Curry*[*exp x a* ⊗ *exp x b*, *x*, *a* ⊗ *b*]
⟨*eval x a* · ⟨$\mathfrak{p}_1$[*exp x a*, *exp x b*] · $\mathfrak{p}_1$[*exp x a* ⊗ *exp x b*, *x*],
$\mathfrak{p}_0$[*exp x a* ⊗ *exp x b*, *x*]⟩,
eval x b · ⟨$\mathfrak{p}_0$[*exp x a*, *exp x b*] · $\mathfrak{p}_1$[*exp x a* ⊗ *exp x b*, *x*],
$\mathfrak{p}_0$[*exp x a* ⊗ *exp x b*, *x*]⟩⟩)
⟨*Curry*[*exp x* (*a* ⊗ *b*), *x*, *a*] ($\mathfrak{p}_1$[*a*, *b*] · *eval x* (*a* ⊗ *b*)),
Curry[*exp x* (*a* ⊗ *b*), *x*, *b*] ($\mathfrak{p}_0$[*a*, *b*] · *eval x* (*a* ⊗ *b*))⟩
and *isomorphic* (*exp x* (*a* ⊗ *b*)) (*exp x a* ⊗ *exp x b*)
proof −
have *A*: «*?A* : (*exp x a* ⊗ *exp x b*) ⊗ *x* → *a*»
using *assms* **by** *auto*
have *B*: «*?B* : (*exp x a* ⊗ *exp x b*) ⊗ *x* → *b*»
using *assms* **by** *auto*
have *AB*: «⟨*?A*, *?B*⟩ : (*exp x a* ⊗ *exp x b*) ⊗ *x* → *a* ⊗ *b*»
by (*metis A B ECC.tuple-in-hom prod-eq-tensor*)
have *C*: «*?C* : *exp x* (*a* ⊗ *b*) → *exp x a*»
using *assms* **by** *auto*

have *D*: «*?D* : *exp x* (*a* ⊗ *b*) → *exp x b*»
  using *assms* **by** *auto*
show *CD*: «⟨*?C*, *?D*⟩ : *exp x* (*a* ⊗ *b*) → *exp x a* ⊗ *exp x b*»
  using *C D* **by** *fastforce*
show *1*: «*Curry*[*exp x a* ⊗ *exp x b*, *x*, *a* ⊗ *b*] ⟨*?A*, *?B*⟩
      : (*exp x a* ⊗ *exp x b*) → *exp x* (*a* ⊗ *b*)»
  by (*simp add*: *AB assms*(*1*−*3*) *Curry-in-hom*)
show *inverse-arrows* (*Curry*[*exp x a* ⊗ *exp x b*, *x*, *a* ⊗ *b*] ⟨*?A*, *?B*⟩) ⟨*?C*, *?D*⟩
proof
  show *ide* (*Curry*[*exp x a* ⊗ *exp x b*, *x*, *a* ⊗ *b*] ⟨*?A*, *?B*⟩ · ⟨*?C*, *?D*⟩)
  proof −
    have *Curry*[*exp x a* ⊗ *exp x b*, *x*, *a* ⊗ *b*] ⟨*?A*, *?B*⟩ · ⟨*?C*, *?D*⟩ =
        Curry[*exp x* (*a* ⊗ *b*), *x*, *a* ⊗ *b*] (⟨*?A*, *?B*⟩ · (⟨*?C*, *?D*⟩ ⊗ *x*))
      using *assms AB CD comp-Curry-arr* **by** *presburger*
    also have ... = *Curry*[*exp x* (*a* ⊗ *b*), *x*, *a* ⊗ *b*]
              ⟨*?A* · (⟨*?C*, *?D*⟩ ⊗ *x*), *?B* · (⟨*?C*, *?D*⟩ ⊗ *x*)⟩
    proof −
      have *span ?A ?B*
        using *A B* **by** *fastforce*
      moreover have *arr* (⟨*?C*, *?D*⟩ ⊗ *x*)
        using *assms CD* **by** *auto*
      moreover have *dom ?A* = *cod* (⟨*?C*, *?D*⟩ ⊗ *x*)
        by (*metis A CD assms*(*3*) *cod-tensor ide-char in-homE*)
      ultimately show *?thesis*
        using *assms ECC.comp-tuple-arr* **by** *metis*
    qed
    also have ... = *Curry*[*exp x* (*a* ⊗ *b*), *x*, *a* ⊗ *b*]
              ⟨*Uncurry*[*x*, *a*] *?C*, *eval x b* · (*?D* ⊗ *x*)⟩
    proof −
      have *?A* · (⟨*?C*, *?D*⟩ ⊗ *x*) = *Uncurry*[*x*, *a*] *?C*
      proof −
        have *?A* · (⟨*?C*, *?D*⟩ ⊗ *x*) =
            eval x a ·
              ⟨$\mathfrak{p}_1$[*exp x a*, *exp x b*] · $\mathfrak{p}_1$[*exp x a* ⊗ *exp x b*, *x*] · (⟨*?C*, *?D*⟩ ⊗ *x*),
               $\mathfrak{p}_0$[*exp x a* ⊗ *exp x b*, *x*] · (⟨*?C*, *?D*⟩ ⊗ *x*)⟩
          using *assms ECC.comp-tuple-arr comp-assoc* **by** *simp*
        also have ... = *eval x a* ·
                  ⟨*?C* · $\mathfrak{p}_1$[*exp x* (*a* ⊗ *b*), *x*], *x* · $\mathfrak{p}_0$[*exp x* (*a* ⊗ *b*), *x*]⟩
          using *assms ECC.pr-naturality*(*1*−*2*) **by** *auto*
        also have ... = *eval x a* · (*?C* ⊗ *x*) ·
                  ⟨$\mathfrak{p}_1$[*exp x* (*a* ⊗ *b*), *x*], $\mathfrak{p}_0$[*exp x* (*a* ⊗ *b*), *x*]⟩
          using *assms*
              ECC.prod-tuple
                [*of* $\mathfrak{p}_1$[*exp x* (*a* ⊗ *b*), *x*] $\mathfrak{p}_0$[*exp x* (*a* ⊗ *b*), *x*] *?C x*]
          by *simp*
        also have ... = *Uncurry*[*x*, *a*] *?C*
          using *assms C ECC.tuple-pr comp-arr-ide comp-arr-dom* **by** *auto*
        finally show *?thesis* **by** *blast*
      qed

moreover have $?B \cdot (\langle ?C, ?D\rangle \otimes x) = Uncurry[x, b]\ ?D$
proof −
have $?B \cdot (\langle ?C, ?D\rangle \otimes x) =$
$eval\ x\ b\ \cdot$
$\langle \mathfrak{p}_0[exp\ x\ a,\ exp\ x\ b] \cdot \mathfrak{p}_1[exp\ x\ a \otimes exp\ x\ b,\ x] \cdot (\langle ?C, ?D\rangle \otimes x),$
$\mathfrak{p}_0[exp\ x\ a \otimes exp\ x\ b,\ x] \cdot (\langle ?C, ?D\rangle \otimes x)\rangle$
using *assms ECC.comp-tuple-arr comp-assoc* **by** *simp*
also have ... $= eval\ x\ b\ \cdot$
$\langle ?D \cdot \mathfrak{p}_1[exp\ x\ (a \otimes b),\ x],\ x \cdot \mathfrak{p}_0[exp\ x\ (a \otimes b),\ x]\rangle$
using *assms C ECC.pr-naturality(1−2)* **by** *auto*
also have ... $= eval\ x\ b \cdot (?D \otimes x)\ \cdot$
$\langle \mathfrak{p}_1[exp\ x\ (a \otimes b),\ x],\ \mathfrak{p}_0[exp\ x\ (a \otimes b),\ x]\rangle$
using *assms*
ECC.prod-tuple
[*of* $\mathfrak{p}_1[exp\ x\ (a \otimes b),\ x]\ \mathfrak{p}_0[exp\ x\ (a \otimes b),\ x]\ ?D\ x$]
by *simp*
also have ... $= Uncurry[x, b]\ ?D$
using *assms C ECC.tuple-pr comp-arr-ide comp-arr-dom* **by** *auto*
finally show *?thesis* **by** *blast*
qed
ultimately show *?thesis* **by** *simp*
qed
also have ... $= Curry[exp\ x\ (a \otimes b),\ x,\ a \otimes b]$
$(\langle \mathfrak{p}_1[a, b] \cdot eval\ x\ (a \otimes b),\ \mathfrak{p}_0[a, b] \cdot eval\ x\ (a \otimes b)\rangle)$
using *assms Uncurry-Curry* **by** *auto*
also have ... $= Curry[exp\ x\ (a \otimes b),\ x,\ a \otimes b]$
$(\langle \mathfrak{p}_1[a, b],\ \mathfrak{p}_0[a, b]\rangle \cdot eval\ x\ (a \otimes b))$
using *assms ECC.comp-tuple-arr* [*of* $\mathfrak{p}_1[a, b]\ \mathfrak{p}_0[a, b]\ eval\ x\ (a \otimes b)$]
by *simp*
also have ... $= Curry[exp\ x\ (a \otimes b),\ x,\ a \otimes b]\ (eval\ x\ (a \otimes b))$
using *assms comp-cod-arr* **by** *simp*
also have ... $= exp\ x\ (a \otimes b)$
using *assms Curry-Uncurry ide-exp ide-in-hom tensor-preserves-ide*
Uncurry-exp
by *metis*
finally have $Curry[exp\ x\ a \otimes exp\ x\ b,\ x,\ a \otimes b]\ \langle ?A, ?B\rangle \cdot \langle ?C, ?D\rangle =$
$exp\ x\ (a \otimes b)$
by *blast*
thus *?thesis*
using *assms ide-exp tensor-preserves-ide* **by** *presburger*
qed
show $ide\ (\langle ?C, ?D\rangle \cdot Curry[exp\ x\ a \otimes exp\ x\ b,\ x,\ a \otimes b]\ \langle ?A, ?B\rangle)$
proof −
have *2*: $span\ \mathfrak{p}_1[exp\ x\ a \otimes exp\ x\ b,\ x]\ \mathfrak{p}_0[exp\ x\ a \otimes exp\ x\ b,\ x]$
using *assms* **by** *simp*
have *3*: $seq\ x\ \mathfrak{p}_0[exp\ x\ a \otimes exp\ x\ b,\ x]$
using *assms* **by** *simp*
have $\langle ?C, ?D\rangle \cdot Curry[exp\ x\ a \otimes exp\ x\ b,\ x,\ a \otimes b]\ \langle ?A, ?B\rangle =$
$\langle ?C \cdot Curry[exp\ x\ a \otimes exp\ x\ b,\ x,\ a \otimes b]\ \langle ?A, ?B\rangle,$

?D · *Curry*[*exp x a* ⊗ *exp x b*, *x*, *a* ⊗ *b*] ⟨*?A*, *?B*⟩⟩
using *assms C D 1 ECC.comp-tuple-arr* **by** (*metis in-homE*)
also have ... = ⟨$\mathfrak{p}_1$[*exp x a*, *exp x b*], $\mathfrak{p}_0$[*exp x a*, *exp x b*]⟩
proof −
have *Curry*[*exp x* (*a* ⊗ *b*), *x*, *a*] ($\mathfrak{p}_1$[*a*, *b*] · *eval x* (*a* ⊗ *b*)) ·
Curry[*exp x a* ⊗ *exp x b*, *x*, *a* ⊗ *b*] ⟨*?A*, *?B*⟩ =
$\mathfrak{p}_1$[*exp x a*, *exp x b*]
proof −
have *Curry*[*exp x* (*a* ⊗ *b*), *x*, *a*] ($\mathfrak{p}_1$[*a*, *b*] · *eval x* (*a* ⊗ *b*)) ·
Curry[*exp x a* ⊗ *exp x b*, *x*, *a* ⊗ *b*] ⟨*?A*, *?B*⟩ =
Curry[*exp x a* ⊗ *exp x b*, *x*, *a*]
(($\mathfrak{p}_1$[*a*, *b*] · *eval x* (*a* ⊗ *b*)) ·
(*Curry*[*exp x a* ⊗ *exp x b*, *x*, *a* ⊗ *b*] ⟨*?A*, *?B*⟩ ⊗ *x*))
using *assms 1 comp-Curry-arr* **by** *auto*
also have ... = *Curry*[*exp x a* ⊗ *exp x b*, *x*, *a*]
($\mathfrak{p}_1$[*a*, *b*] · *eval x* (*a* ⊗ *b*) ·
(*Curry*[*exp x a* ⊗ *exp x b*, *x*, *a* ⊗ *b*] ⟨*?A*, *?B*⟩ ⊗ *x*))
using *comp-assoc* **by** *simp*
also have ... = *Curry*[*exp x a* ⊗ *exp x b*, *x*, *a*] ($\mathfrak{p}_1$[*a*, *b*] · ⟨*?A*, *?B*⟩)
using *assms AB Uncurry-Curry ide-exp tensor-preserves-ide* **by** *simp*
also have ... = *Curry*[*exp x a* ⊗ *exp x b*, *x*, *a*]
(*eval x a* ·
⟨$\mathfrak{p}_1$[*exp x a*, *exp x b*] · $\mathfrak{p}_1$[*exp x a* ⊗ *exp x b*, *x*],
$\mathfrak{p}_0$[*exp x a* ⊗ *exp x b*, *x*]⟩)
using *assms A B ECC.pr-tuple*(*1*) **by** *fastforce*
also have ... = *Curry*[*exp x a* ⊗ *exp x b*, *x*, *a*]
(*eval x a* · ($\mathfrak{p}_1$[*exp x a*, *exp x b*] ⊗ *x*) ·
⟨$\mathfrak{p}_1$[*exp x a* ⊗ *exp x b*, *x*],
$\mathfrak{p}_0$[*exp x a* ⊗ *exp x b*, *x*]⟩)
proof −
have *seq* $\mathfrak{p}_1$[*exp x a*, *exp x b*] $\mathfrak{p}_1$[*exp x a* ⊗ *exp x b*, *x*]
using *assms* **by** *auto*
thus *?thesis*
using *assms 2 3 prod-eq-tensor comp-ide-arr ECC.prod-tuple*
by *metis*
qed
also have ... = *Curry* (*exp x a* ⊗ *exp x b*) *x a*
(*eval x a* · ($\mathfrak{p}_1$[*exp x a*, *exp x b*] ⊗ *x*))
using *assms comp-arr-dom* **by** *simp*
also have ... = $\mathfrak{p}_1$[*exp x a*, *exp x b*]
using *assms Curry-Uncurry* **by** *simp*
finally show *?thesis* **by** *blast*
qed
moreover have *Curry*[*exp x* (*a* ⊗ *b*), *x*, *b*] ($\mathfrak{p}_0$[*a*, *b*] · *eval x* (*a* ⊗ *b*)) ·
Curry[*exp x a* ⊗ *exp x b*, *x*, *a* ⊗ *b*] ⟨*?A*, *?B*⟩ =
$\mathfrak{p}_0$[*exp x a*, *exp x b*]
proof −
have *Curry*[*exp x* (*a* ⊗ *b*), *x*, *b*] ($\mathfrak{p}_0$[*a*, *b*] · *eval x* (*a* ⊗ *b*)) ·
Curry[*exp x a* ⊗ *exp x b*, *x*, *a* ⊗ *b*] ⟨*?A*, *?B*⟩ =

$Curry[exp\ x\ a \otimes exp\ x\ b,\ x,\ b]$
$((\mathfrak{p}_0[a,\ b] \cdot eval\ x\ (a \otimes b)) \cdot$
$(Curry[exp\ x\ a \otimes exp\ x\ b,\ x,\ a \otimes b]\ \langle ?A,\ ?B\rangle \otimes x))$
proof −
have «$Curry[exp\ x\ a \otimes exp\ x\ b,\ x,\ a \otimes b]\ \langle ?A,\ ?B\rangle$
$: exp\ x\ a \otimes exp\ x\ b \rightarrow exp\ x\ (a \otimes b)$»
using *1* **by** *blast*
moreover have «$\mathfrak{p}_0[a,\ b] \cdot eval\ x\ (a \otimes b) : exp\ x\ (a \otimes b) \otimes x \rightarrow b$»
using *assms*
by (*metis* (*no-types*, *lifting*) *ECC.pr0-in-hom′ ECC.pr-simps*(*2*)
comp-in-homI $eval\text{-}in\text{-}hom_{ECMC}$ *prod-eq-tensor tensor-preserves-ide*)
ultimately show *?thesis*
using *assms comp-Curry-arr* **by** *simp*
qed
also have ... = $Curry[exp\ x\ a \otimes exp\ x\ b,\ x,\ b]$
$(\mathfrak{p}_0[a,\ b] \cdot$
$Uncurry[x,\ a \otimes b]$
$(Curry[exp\ x\ a \otimes exp\ x\ b,\ x,\ a \otimes b]\ \langle ?A,\ ?B\rangle))$
using *comp-assoc* **by** *simp*
also have ... = $Curry\ (exp\ x\ a \otimes exp\ x\ b)\ x\ b\ (\mathfrak{p}_0[a,\ b] \cdot \langle ?A,\ ?B\rangle)$
using *assms AB Uncurry-Curry ide-exp tensor-preserves-ide* **by** *simp*
also have ... = $Curry[exp\ x\ a \otimes exp\ x\ b,\ x,\ b]$
$(eval\ x\ b \cdot$
$\langle \mathfrak{p}_0[exp\ x\ a,\ exp\ x\ b] \cdot \mathfrak{p}_1[exp\ x\ a \otimes exp\ x\ b,\ x],$
$\mathfrak{p}_0[exp\ x\ a \otimes exp\ x\ b,\ x]\rangle)$
using *assms A B* **by** *fastforce*
also have ... = $Curry[exp\ x\ a \otimes exp\ x\ b,\ x,\ b]$
$(eval\ x\ b \cdot (\mathfrak{p}_0[exp\ x\ a,\ exp\ x\ b] \otimes x) \cdot$
$\langle \mathfrak{p}_1[exp\ x\ a \otimes exp\ x\ b,\ x],$
$\mathfrak{p}_0[exp\ x\ a \otimes exp\ x\ b,\ x]\rangle)$
proof −
have $seq\ \mathfrak{p}_0[exp\ x\ a,\ exp\ x\ b]\ \mathfrak{p}_1[exp\ x\ a \otimes exp\ x\ b,\ x]$
using *assms* **by** *auto*
thus *?thesis*
using *assms 2 3 prod-eq-tensor comp-ide-arr ECC.prod-tuple*
by *metis*
qed
also have ... = $Curry\ (exp\ x\ a \otimes exp\ x\ b)\ x\ b$
$(Uncurry[x,\ b]\ \mathfrak{p}_0[exp\ x\ a,\ exp\ x\ b])$
proof −
have $(\mathfrak{p}_0[exp\ x\ a,\ exp\ x\ b] \otimes x) \cdot$
$\langle \mathfrak{p}_1[exp\ x\ a \otimes exp\ x\ b,\ x],\ \mathfrak{p}_0[exp\ x\ a \otimes exp\ x\ b,\ x]\rangle =$
$\mathfrak{p}_0[exp\ x\ a,\ exp\ x\ b] \otimes x$
using *assms comp-arr-ide ECC.tuple-pr* **by** *auto*
thus *?thesis* **by** *simp*
qed
also have ... = $\mathfrak{p}_0[exp\ x\ a,\ exp\ x\ b]$
using *assms Curry-Uncurry* **by** *simp*
finally show *?thesis* **by** *blast*

qed
ultimately show *?thesis* **by** *simp*
qed
also have ... = *exp x a* ⊗ *exp x b*
using *assms ECC.tuple-pr* **by** *simp*
finally have ⟨*?C*, *?D*⟩ · *Curry*[*exp x a* ⊗ *exp x b*, *x*, *a* ⊗ *b*] ⟨*?A*, *?B*⟩ =
exp x a ⊗ *exp x b*
by *blast*
thus *?thesis*
using *assms tensor-preserves-ide* **by** *simp*
qed
qed
thus *isomorphic* (*exp x* (*a* ⊗ *b*)) (*exp x a* ⊗ *exp x b*)
unfolding *isomorphic-def*
using *CD* **by** *blast*
qed

end

end

Chapter 2

Enriched Categories

The notion of an enriched category [1] generalizes the concept of category by replacing the hom-sets of an ordinary category by objects of an arbitrary monoidal category *M*. The choice, for each object *a*, of a distinguished element *id a* : *a* → *a* as an identity, is replaced by an arrow *Id a* : $\mathcal{I}$ → *Hom a a* of *M*. The composition operation is similarly replaced by a family of arrows *Comp a b c* : *Hom B C* ⊗ *Hom A B* → *Hom A C* of *M*. The identity and composition are required to satisfy unit and associativity laws which are expressed as commutative diagrams in *M*.

theory *EnrichedCategory*
imports *ClosedMonoidalCategory*
begin

context *monoidal-category*
begin

abbreviation ι' (ι^{-1})
where $\iota' \equiv$ *inv* ι

end

context *elementary-symmetric-monoidal-category*
begin

lemma *sym-unit*:
shows $\iota \cdot \mathrm{s}[\mathcal{I}, \mathcal{I}] = \iota$
by (*simp add*: *ι-def unitor-coherence unitor-coincidence*(*2*))

lemma *sym-inv-unit*:
shows $\mathrm{s}[\mathcal{I}, \mathcal{I}] \cdot$ *inv* ι = *inv* ι
using *sym-unit*
by (*metis MC.unit-is-iso arr-inv cod-inv comp-arr-dom comp-cod-arr*

iso-cancel-left iso-is-arr)

end

2.1 Basic Definitions

locale *enriched-category* =
monoidal-category +
fixes *Obj* :: *'o set*
and *Hom* :: *'o* ⇒ *'o* ⇒ *'a*
and *Id* :: *'o* ⇒ *'a*
and *Comp* :: *'o* ⇒ *'o* ⇒ *'o* ⇒ *'a*
assumes *ide-Hom* [*intro*, *simp*]: ⟦$a \in Obj$; $b \in Obj$⟧ ⟹ *ide* (*Hom a b*)
and *Id-in-hom* [*intro*]: $a \in Obj$ ⟹ «*Id a* : $\mathcal{I}$ → *Hom a a*»
and *Comp-in-hom* [*intro*]: ⟦$a \in Obj$; $b \in Obj$; $c \in Obj$⟧ ⟹
«*Comp a b c* : *Hom b c* ⊗ *Hom a b* → *Hom a c*»
and *Comp-Hom-Id*: ⟦$a \in Obj$; $b \in Obj$⟧ ⟹
Comp a a b · (*Hom a b* ⊗ *Id a*) = r[*Hom a b*]
and *Comp-Id-Hom*: ⟦$a \in Obj$; $b \in Obj$⟧ ⟹
Comp a b b · (*Id b* ⊗ *Hom a b*) = l[*Hom a b*]
and *Comp-assoc*: ⟦$a \in Obj$; $b \in Obj$; $c \in Obj$; $d \in Obj$⟧ ⟹
Comp a b d · (*Comp b c d* ⊗ *Hom a b*) =
Comp a c d · (*Hom c d* ⊗ *Comp a b c*) ·
a[*Hom c d*, *Hom b c*, *Hom a b*]

A functor from an enriched category A to an enriched category B consists of an object map $F_o : Obj_A \to Obj_B$ and a map F_a that assigns to each pair of objects $a\ b$ in Obj_A an arrow $F_a\ a\ b : Hom_A\ a\ b \to Hom_B\ (F_o\ a)\ (F_o\ b)$ of the underlying monoidal category, subject to equations expressing that identities and composition are preserved.

locale *enriched-functor* =
monoidal-category C T α ι +
A: *enriched-category C T α ι* Obj_A Hom_A Id_A $Comp_A$ +
B: *enriched-category C T α ι* Obj_B Hom_B Id_B $Comp_B$
for *C* :: *'m* ⇒ *'m* ⇒ *'m* (**infixr** ‹·› *55*)
and *T* :: *'m* × *'m* ⇒ *'m*
and α :: *'m* × *'m* × *'m* ⇒ *'m*
and ι :: *'m*
and Obj_A :: *'a set*
and Hom_A :: *'a* ⇒ *'a* ⇒ *'m*
and Id_A :: *'a* ⇒ *'m*
and $Comp_A$ :: *'a* ⇒ *'a* ⇒ *'a* ⇒ *'m*
and Obj_B :: *'b set*
and Hom_B :: *'b* ⇒ *'b* ⇒ *'m*
and Id_B :: *'b* ⇒ *'m*
and $Comp_B$ :: *'b* ⇒ *'b* ⇒ *'b* ⇒ *'m*
and F_o :: *'a* ⇒ *'b*
and F_a :: *'a* ⇒ *'a* ⇒ *'m* +

assumes *extensionality*: $a \notin Obj_A \vee b \notin Obj_A \Longrightarrow F_a\ a\ b = null$
assumes *preserves-Obj* [*intro*]: $a \in Obj_A \Longrightarrow F_o\ a \in Obj_B$
and *preserves-Hom*: $[\![a \in Obj_A;\ b \in Obj_A]\!] \Longrightarrow$
$«F_a\ a\ b : Hom_A\ a\ b \rightarrow Hom_B\ (F_o\ a)\ (F_o\ b)»$
and *preserves-Id*: $a \in Obj_A \Longrightarrow F_a\ a\ a \cdot Id_A\ a = Id_B\ (F_o\ a)$
and *preserves-Comp*: $[\![a \in Obj_A;\ b \in Obj_A;\ c \in Obj_A]\!] \Longrightarrow$
$Comp_B\ (F_o\ a)\ (F_o\ b)\ (F_o\ c) \cdot T\ (F_a\ b\ c,\ F_a\ a\ b) =$
$F_a\ a\ c \cdot Comp_A\ a\ b\ c$

locale *fully-faithful-enriched-functor* =
enriched-functor +
assumes *locally-iso*: $[\![a \in Obj_A;\ b \in Obj_A]\!] \Longrightarrow iso\ (F_a\ a\ b)$

A natural transformation from an an enriched functor $F = (F_o,\ F_a)$ to an enriched functor $G = (G_o,\ G_a)$ consists of a map τ that assigns to each object $a \in Obj_A$ a "component at a", which is an arrow $\tau\ a : \mathcal{I} \rightarrow Hom_B\ (F_o\ a)\ (G_o\ a)$, subject to an equation that expresses the naturality condition.

locale *enriched-natural-transformation* =
monoidal-category $C\ T\ \alpha\ \iota$ +
A: *enriched-category* $C\ T\ \alpha\ \iota\ Obj_A\ Hom_A\ Id_A\ Comp_A$ +
B: *enriched-category* $C\ T\ \alpha\ \iota\ Obj_B\ Hom_B\ Id_B\ Comp_B$ +
F: *enriched-functor* $C\ T\ \alpha\ \iota$
$Obj_A\ Hom_A\ Id_A\ Comp_A\ Obj_B\ Hom_B\ Id_B\ Comp_B\ F_o\ F_a$ +
G: *enriched-functor* $C\ T\ \alpha\ \iota$
$Obj_A\ Hom_A\ Id_A\ Comp_A\ Obj_B\ Hom_B\ Id_B\ Comp_B\ G_o\ G_a$
for $C :: 'm \Rightarrow 'm \Rightarrow 'm$ (**infixr** ‹·› *55*)
and $T :: 'm \times 'm \Rightarrow 'm$
and $\alpha :: 'm \times 'm \times 'm \Rightarrow 'm$
and $\iota :: 'm$
and $Obj_A :: 'a\ set$
and $Hom_A :: 'a \Rightarrow 'a \Rightarrow 'm$
and $Id_A :: 'a \Rightarrow 'm$
and $Comp_A :: 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'm$
and $Obj_B :: 'b\ set$
and $Hom_B :: 'b \Rightarrow 'b \Rightarrow 'm$
and $Id_B :: 'b \Rightarrow 'm$
and $Comp_B :: 'b \Rightarrow 'b \Rightarrow 'b \Rightarrow 'm$
and $F_o :: 'a \Rightarrow 'b$
and $F_a :: 'a \Rightarrow 'a \Rightarrow 'm$
and $G_o :: 'a \Rightarrow 'b$
and $G_a :: 'a \Rightarrow 'a \Rightarrow 'm$
and $\tau :: 'a \Rightarrow 'm$ +
assumes *extensionality*: $a \notin Obj_A \Longrightarrow \tau\ a = null$
and *component-in-hom* [*intro*]: $a \in Obj_A \Longrightarrow «\tau\ a : \mathcal{I} \rightarrow Hom_B\ (F_o\ a)\ (G_o\ a)»$
and *naturality*: $[\![a \in Obj_A;\ b \in Obj_A]\!] \Longrightarrow$
$Comp_B\ (F_o\ a)\ (F_o\ b)\ (G_o\ b) \cdot (\tau\ b \otimes F_a\ a\ b) \cdot \mathrm{l}^{-1}[Hom_A\ a\ b] =$
$Comp_B\ (F_o\ a)\ (G_o\ a)\ (G_o\ b) \cdot (G_a\ a\ b \otimes \tau\ a) \cdot \mathrm{r}^{-1}[Hom_A\ a\ b]$

2.1.1 Self-Enrichment

context *elementary-closed-monoidal-category*
begin

Every closed monoidal category M admits a structure of enriched category, where the exponentials in M itself serve as the "hom-objects" (*cf.* [1] Section 1.6). Essentially all the work in proving this theorem has already been done in *EnrichedCategoryBasics.ClosedMonoidalCategory*.

interpretation *closed-monoidal-category*
using *is-closed-monoidal-category* **by** *blast*

interpretation *EC*: *enriched-category C T* α ι ‹*Collect ide*› *exp Id Comp*
using *Id-in-hom Comp-in-hom Comp-unit-right Comp-unit-left* $Comp\text{-}assoc_{ECMC}(2)$
by *unfold-locales auto*

theorem *is-enriched-in-itself*:
shows *enriched-category C T* α ι (*Collect ide*) *exp Id Comp*
..

The following mappings define a bijection between *hom a b* and *hom* $\mathcal{I}$ (*exp a b*). These have functorial properties which are encountered repeatedly.

definition *UP* ($\text{-}^{\uparrow}$ [*100*] *100*)
where $t^{\uparrow} \equiv$ *if arr t then* $Curry[\mathcal{I}, dom\ t, cod\ t]\ (t \cdot \mathrm{l}[dom\ t])$ *else null*

definition *DN*
where *DN a b t* $\equiv$ *if arr t then* $Uncurry[a, b]\ t \cdot \mathrm{l}^{-1}[a]$ *else null*

abbreviation DN' ($\text{-}^{\downarrow}[\text{-}, \text{-}]$ [*100*] *99*)
where $t^{\downarrow}[a, b] \equiv$ *DN a b t*

lemma *UP-DN*:
shows [*intro*]: $arr\ t \Longrightarrow «t^{\uparrow} : \mathcal{I} \rightarrow exp\ (dom\ t)\ (cod\ t)»$
and [*intro*]: $⟦ide\ a;\ ide\ b;\ «t : \mathcal{I} \rightarrow exp\ a\ b»⟧ \Longrightarrow «t^{\downarrow}[a, b]: a \rightarrow b»$
and [*simp*]: $arr\ t \Longrightarrow (t^{\uparrow})^{\downarrow}[dom\ t, cod\ t] = t$
and [*simp*]: $⟦ide\ a;\ ide\ b;\ «t : \mathcal{I} \rightarrow exp\ a\ b»⟧ \Longrightarrow (t^{\downarrow}[a, b])^{\uparrow} = t$
using *UP-def DN-def Uncurry-Curry Curry-Uncurry* [*of* $\mathcal{I}$ *a b t*]
comp-assoc comp-arr-dom
by *auto*

lemma *UP-simps* [*simp*]:
assumes *arr t*
shows *arr* $(t^{\uparrow})$ **and** *dom* $(t^{\uparrow}) = \mathcal{I}$ **and** *cod* $(t^{\uparrow}) = exp\ (dom\ t)\ (cod\ t)$
using *assms UP-DN* **by** *auto*

lemma *DN-simps* [*simp*]:
assumes *ide a* **and** *ide b* **and** *arr t* **and** *dom t* $= \mathcal{I}$ **and** *cod t* $= exp\ a\ b$
shows *arr* $(t^{\downarrow}[a, b])$ **and** *dom* $(t^{\downarrow}[a, b]) = a$ **and** *cod* $(t^{\downarrow}[a, b]) = b$

using *assms UP-DN DN-def* **by** *auto*

lemma *UP-ide*:
assumes *ide a*
shows $a^{\uparrow} = Id\ a$
using *assms Id-def comp-cod-arr UP-def* **by** *auto*

lemma *DN-Id*:
assumes *ide a*
shows $(Id\ a)^{\downarrow}[a,\ a] = a$
using *assms Uncurry-Curry lunit-in-hom Id-def DN-def* **by** *auto*

lemma *UP-comp*:
assumes *seq t u*
shows $(t \cdot u)^{\uparrow} = Comp\ (dom\ u)\ (cod\ u)\ (cod\ t) \cdot (t^{\uparrow} \otimes u^{\uparrow}) \cdot \iota^{-1}$
proof –
have $Comp\ (dom\ u)\ (cod\ u)\ (cod\ t) \cdot (t^{\uparrow} \otimes u^{\uparrow}) \cdot \iota^{-1} =$
$(Curry[exp\ (cod\ u)\ (cod\ t) \otimes exp\ (dom\ u)\ (cod\ u),\ dom\ u,\ cod\ t]$
$(eval\ (cod\ u)\ (cod\ t) \cdot$
$(exp\ (cod\ u)\ (cod\ t) \otimes eval\ (dom\ u)\ (cod\ u)) \cdot$
$\mathrm{a}[exp\ (cod\ u)\ (cod\ t),\ exp\ (dom\ u)\ (cod\ u),\ dom\ u]) \cdot$
$(t^{\uparrow} \otimes u^{\uparrow})) \cdot \iota^{-1}$
unfolding *Comp-def*
using *assms comp-assoc* **by** *simp*
also have ... =
$(Curry[\mathcal{I} \otimes \mathcal{I},\ dom\ u,\ cod\ t]$
$((eval\ (cod\ u)\ (cod\ t) \cdot$
$(exp\ (cod\ u)\ (cod\ t) \otimes eval\ (dom\ u)\ (cod\ u)) \cdot$
$\mathrm{a}[exp\ (cod\ u)\ (cod\ t),\ exp\ (dom\ u)\ (cod\ u),\ dom\ u]) \cdot$
$((t^{\uparrow} \otimes u^{\uparrow}) \otimes dom\ u))) \cdot \iota^{-1}$
using *assms*
comp-Curry-arr
$[of\ dom\ u\ t^{\uparrow} \otimes u^{\uparrow}$
$\mathcal{I} \otimes \mathcal{I}\ exp\ (cod\ u)\ (cod\ t) \otimes exp\ (dom\ u)\ (cod\ u)$
$eval\ (cod\ u)\ (cod\ t) \cdot$
$(exp\ (cod\ u)\ (cod\ t) \otimes eval\ (dom\ u)\ (cod\ u)) \cdot$
$\mathrm{a}[exp\ (cod\ u)\ (cod\ t),\ exp\ (dom\ u)\ (cod\ u),\ dom\ u]$
$cod\ t]$
by *fastforce*
also have ... =
$Curry[\mathcal{I} \otimes \mathcal{I},\ dom\ u,\ cod\ t]$
$(eval\ (cod\ u)\ (cod\ t) \cdot$
$((exp\ (cod\ u)\ (cod\ t) \otimes eval\ (dom\ u)\ (cod\ u)) \cdot$
$(t^{\uparrow} \otimes u^{\uparrow} \otimes dom\ u)) \cdot \mathrm{a}[\mathcal{I},\ \mathcal{I},\ dom\ u]) \cdot \iota^{-1}$
using *assms assoc-naturality* $[of\ t^{\uparrow}\ u^{\uparrow}\ dom\ u]$ *comp-assoc* **by** *auto*
also have ... =
$Curry[\mathcal{I} \otimes \mathcal{I},\ dom\ u,\ cod\ t]$
$(eval\ (cod\ u)\ (cod\ t) \cdot$
$(exp\ (cod\ u)\ (cod\ t) \cdot t^{\uparrow} \otimes Uncurry[dom\ u,\ cod\ u]\ (u^{\uparrow})) \cdot$

$\mathrm{a}[\mathcal{I}, \mathcal{I}, dom\ u]) \cdot \iota^{-1}$
using *assms comp-cod-arr UP-DN interchange* **by** *auto*
also have ... =
$Curry[\mathcal{I} \otimes \mathcal{I}, dom\ u, cod\ t]$
$(eval\ (cod\ u)\ (cod\ t) \cdot$
$(exp\ (cod\ u)\ (cod\ t) \cdot t^{\uparrow} \otimes u \cdot \mathrm{l}[dom\ u]) \cdot$
$\mathrm{a}[\mathcal{I}, \mathcal{I}, dom\ u]) \cdot \iota^{-1}$
using *assms Uncurry-Curry UP-def* **by** *auto*
also have ... =
$Curry[\mathcal{I} \otimes \mathcal{I}, dom\ u, cod\ t]$
$(eval\ (cod\ u)\ (cod\ t) \cdot$
$(t^{\uparrow} \otimes u \cdot \mathrm{l}[dom\ u]) \cdot \mathrm{a}[\mathcal{I}, \mathcal{I}, dom\ u]) \cdot \iota^{-1}$
using *assms comp-cod-arr* **by** *auto*
also have ... =
$Curry[\mathcal{I} \otimes \mathcal{I}, dom\ u, cod\ t]$
$(eval\ (cod\ u)\ (cod\ t) \cdot$
$((t^{\uparrow} \otimes cod\ u) \cdot (\mathcal{I} \otimes u \cdot \mathrm{l}[dom\ u])) \cdot$
$\mathrm{a}[\mathcal{I}, \mathcal{I}, dom\ u]) \cdot \iota^{-1}$
using *assms comp-arr-dom* [*of* $t^{\uparrow}$ $\mathcal{I}$] *comp-cod-arr* [*of* $u \cdot \mathrm{l}[dom\ u]$ *cod u*]
interchange [*of* $t^{\uparrow}$ $\mathcal{I}$ *cod u u* $\cdot\ \mathrm{l}[dom\ u]$]
by *auto*
also have ... =
$Curry[\mathcal{I}, dom\ u, cod\ t]$
$((eval\ (cod\ u)\ (cod\ t) \cdot$
$((t^{\uparrow} \otimes cod\ u) \cdot (\mathcal{I} \otimes u \cdot \mathrm{l}[dom\ u])) \cdot$
$\mathrm{a}[\mathcal{I}, \mathcal{I}, dom\ u]) \cdot (\iota^{-1} \otimes dom\ u))$
proof −
have «$\iota^{-1} : \mathcal{I} \rightarrow \mathcal{I} \otimes \mathcal{I}$»
using *inv-in-hom unit-is-iso* **by** *blast*
thus *?thesis*
using *assms comp-Curry-arr* **by** *fastforce*
qed
also have ... =
$Curry[\mathcal{I}, dom\ u, cod\ t]$
$((Uncurry[cod\ u, cod\ t]\ (t^{\uparrow})) \cdot (\mathcal{I} \otimes u \cdot \mathrm{l}[dom\ u]) \cdot$
$\mathrm{a}[\mathcal{I}, \mathcal{I}, dom\ u] \cdot (\iota^{-1} \otimes dom\ u))$
using *comp-assoc* **by** *simp*
also have ... = $Curry[\mathcal{I}, dom\ u, cod\ t]\ (Uncurry[cod\ u, cod\ t]\ (t^{\uparrow}) \cdot (\mathcal{I} \otimes u))$
proof −
have $(\mathcal{I} \otimes u \cdot \mathrm{l}[dom\ u]) \cdot \mathrm{a}[\mathcal{I}, \mathcal{I}, dom\ u] \cdot (\iota^{-1} \otimes dom\ u) =$
$((\mathcal{I} \otimes u) \cdot (\mathcal{I} \otimes \mathrm{l}[dom\ u])) \cdot \mathrm{a}[\mathcal{I}, \mathcal{I}, dom\ u] \cdot (\iota^{-1} \otimes dom\ u)$
using *assms* **by** *auto*
also have ... = $(\mathcal{I} \otimes u) \cdot (\mathcal{I} \otimes \mathrm{l}[dom\ u]) \cdot \mathrm{a}[\mathcal{I}, \mathcal{I}, dom\ u] \cdot (\iota^{-1} \otimes dom\ u)$
using *comp-assoc* **by** *simp*
also have ... = $(\mathcal{I} \otimes u) \cdot (\mathcal{I} \otimes \mathrm{l}[dom\ u]) \cdot (\mathcal{I} \otimes \mathrm{l}^{-1}[dom\ u])$
proof −
have $\mathrm{a}[\mathcal{I}, \mathcal{I}, dom\ u] \cdot (\iota^{-1} \otimes dom\ u) = \mathcal{I} \otimes \mathrm{l}^{-1}[dom\ u]$
proof −
have $\mathrm{a}[\mathcal{I}, \mathcal{I}, dom\ u] \cdot (\iota^{-1} \otimes dom\ u) =$

```
        inv ((ι ⊗ dom u) · a⁻¹[𝓘, 𝓘, dom u])
      using assms inv-inv inv-comp [of a⁻¹[𝓘, 𝓘, dom u] ι ⊗ dom u]
        inv-tensor [of ι dom u]
      by (metis ide-dom ide-is-iso ide-unity inv-ide iso-assoc iso-inv-iso
          iso-is-arr lunit-char(2) seqE tensor-preserves-iso triangle
          unit-is-iso unitor-coincidence(2))
    also have ... = inv (𝓘 ⊗ l[dom u])
      using assms lunit-char [of dom u] by auto
    also have ... = 𝓘 ⊗ l⁻¹[dom u]
      using assms inv-tensor by auto
    finally show ?thesis by blast
  qed
  thus ?thesis by simp
qed
also have ... = (𝓘 ⊗ u) · (𝓘 ⊗ dom u)
  using assms
  by (metis comp-ide-self comp-lunit-lunit'(1) dom-comp ideD(1)
      ide-dom ide-unity interchange)
also have ... = 𝓘 ⊗ u
  using assms by blast
finally have (𝓘 ⊗ u · l[dom u]) · a[𝓘, 𝓘, dom u] · (ι⁻¹ ⊗ dom u) = 𝓘 ⊗ u
  by blast
thus ?thesis by argo
qed
also have ... = Curry[𝓘, dom u, cod t] ((t · l[cod u]) · (𝓘 ⊗ u))
  using assms Uncurry-Curry UP-def by auto
also have ... = Curry[𝓘, dom u, cod t] (t · u · l[dom u])
  using assms comp-assoc lunit-naturality by auto
also have ... = (t · u)↑
  using assms comp-assoc UP-def by simp
finally show ?thesis by simp
qed

end
```

2.2 Underlying Category, Functor, and Natural Transformation

2.2.1 Underlying Category

The underlying category (*cf.* [1] Section 1.3) of an enriched category has as its arrows from a to b the arrows $\mathcal{I} \rightarrow Hom\ a\ b$ of M (*i.e.* the points of $Hom\ a\ b$). The identity at a is $Id\ a$. The composition of arrows f and g is given by the formula: $Comp\ a\ b\ c \cdot (g \otimes f) \cdot \iota^{-1}$.

```
locale underlying-category =
  M: monoidal-category +
  A: enriched-category
begin
```

```
sublocale concrete-category Obj ‹λa b. M.hom ℐ (Hom a b)› ‹Id›
          ‹λc b a g f. Comp a b c · (g ⊗ f) · ι⁻¹›
proof
  show ⋀a. a ∈ Obj ⟹ Id a ∈ M.hom ℐ (Hom a a)
    using A.Id-in-hom by blast
  show 1: ⋀a b c f g.
           ⟦a ∈ Obj; b ∈ Obj; c ∈ Obj;
            f ∈ M.hom ℐ (Hom a b); g ∈ M.hom ℐ (Hom b c)⟧
               ⟹ Comp a b c · (g ⊗ f) · ι⁻¹ ∈ M.hom ℐ (Hom a c)
    using A.Comp-in-hom M.inv-in-hom M.unit-is-iso M.comp-in-homI
          M.unit-in-hom
    apply auto[1]
    apply (intro M.comp-in-homI)
    by auto
  show ⋀a b f. ⟦a ∈ Obj; b ∈ Obj; f ∈ M.hom ℐ (Hom a b)⟧
                 ⟹ Comp a a b · (f ⊗ Id a) · ι⁻¹ = f
  proof −
    fix a b f
    assume a: a ∈ Obj and b: b ∈ Obj and f: f ∈ M.hom ℐ (Hom a b)
    show Comp a a b · (f ⊗ Id a) · ι⁻¹ = f
    proof −
      have Comp a a b · (f ⊗ Id a) · ι⁻¹ = (Comp a a b · (f ⊗ Id a)) · ι⁻¹
        using M.comp-assoc by simp
      also have ... = (Comp a a b · (Hom a b ⊗ Id a) · (f ⊗ ℐ)) · ι⁻¹
        using a f M.comp-arr-dom M.comp-cod-arr A.Id-in-hom
              M.in-homE M.interchange mem-Collect-eq
        by metis
      also have ... = (r[Hom a b] · (f ⊗ ℐ)) · ι⁻¹
        using a b f A.Comp-Hom-Id M.comp-assoc by metis
      also have ... = (f · r[ℐ]) · ι⁻¹
        using f M.runit-naturality by fastforce
      also have ... = f · ι · ι⁻¹
        by (simp add: M.unitor-coincidence(2) M.comp-assoc)
      also have ... = f
        using f M.comp-arr-dom M.comp-arr-inv' M.unit-is-iso by auto
      finally show Comp a a b · (f ⊗ Id a) · ι⁻¹ = f by blast
    qed
  qed
  show ⋀a b f. ⟦a ∈ Obj; b ∈ Obj; f ∈ M.hom ℐ (Hom a b)⟧
                 ⟹ Comp a b b · (Id b ⊗ f) · ι⁻¹ = f
  proof −
    fix a b f
    assume a: a ∈ Obj and b: b ∈ Obj and f: f ∈ M.hom ℐ (Hom a b)
    show Comp a b b · (Id b ⊗ f) · ι⁻¹ = f
    proof −
      have Comp a b b · (Id b ⊗ f) · ι⁻¹ = (Comp a b b · (Id b ⊗ f)) · ι⁻¹
        using M.comp-assoc by simp
      also have ... = (Comp a b b · (Id b ⊗ Hom a b) · (ℐ ⊗ f)) · ι⁻¹
```

```
      using a b f M.comp-arr-dom M.comp-cod-arr A.Id-in-hom
          M.in-homE M.interchange mem-Collect-eq
      by metis
    also have ... = (l[Hom a b] · (𝓘 ⊗ f)) · ι⁻¹
      using a b A.Comp-Id-Hom M.comp-assoc by metis
    also have ... = (f · l[𝓘]) · ι⁻¹
      using a b f M.lunit-naturality [of f] by auto
    also have ... = f · ι · ι⁻¹
      by (simp add: M.unitor-coincidence(1) M.comp-assoc)
    also have ... = f
      using M.comp-arr-dom M.comp-arr-inv' M.unit-is-iso f by auto
    finally show Comp a b b · (Id b ⊗ f) · ι⁻¹ = f by blast
  qed
qed
show ⋀a b c d f g h.
         ⟦a ∈ Obj; b ∈ Obj; c ∈ Obj; d ∈ Obj;
          f ∈ M.hom 𝓘 (Hom a b); g ∈ M.hom 𝓘 (Hom b c);
          h ∈ M.hom 𝓘 (Hom c d)⟧
            ⟹ Comp a c d · (h ⊗ Comp a b c · (g ⊗ f) · ι⁻¹) · ι⁻¹ =
                Comp a b d · (Comp b c d · (h ⊗ g) · ι⁻¹ ⊗ f) · ι⁻¹
proof -
  fix a b c d f g h
  assume a: a ∈ Obj and b: b ∈ Obj and c: c ∈ Obj and d: d ∈ Obj
  assume f: f ∈ M.hom 𝓘 (Hom a b) and g: g ∈ M.hom 𝓘 (Hom b c)
  and h: h ∈ M.hom 𝓘 (Hom c d)
  have Comp a c d · (h ⊗ Comp a b c · (g ⊗ f) · ι⁻¹) · ι⁻¹ =
        Comp a c d ·
          ((Hom c d ⊗ Comp a b c) · (h ⊗ (g ⊗ f) · ι⁻¹)) · ι⁻¹
 using a b c d f g h 1 M.interchange A.ide-Hom M.comp-ide-arr M.comp-cod-arr
          M.in-homE mem-Collect-eq
    by metis
  also have ... = Comp a c d ·
                    ((Hom c d ⊗ Comp a b c) ·
                        (a[Hom c d, Hom b c, Hom a b] ·
                           a⁻¹[Hom c d, Hom b c, Hom a b])) ·
                        (h ⊗ (g ⊗ f) · ι⁻¹) · ι⁻¹
  proof -
    have (Hom c d ⊗ Comp a b c) ·
            (a[Hom c d, Hom b c, Hom a b] ·
               a⁻¹[Hom c d, Hom b c, Hom a b]) =
          Hom c d ⊗ Comp a b c
      using a b c d
      by (metis A.Comp-in-hom A.ide-Hom M.comp-arr-ide
          M.comp-assoc-assoc'(1) M.ide-in-hom M.interchange M.seqI'
          M.tensor-preserves-ide)
    thus ?thesis
      using M.comp-assoc by force
  qed
  also have ... = (Comp a c d · (Hom c d ⊗ Comp a b c) ·
```

a[*Hom c d*, *Hom b c*, *Hom a b*]) ·
(a^{-1}[*Hom c d*, *Hom b c*, *Hom a b*] ·
(*h* ⊗ (*g* ⊗ *f*) · ι^{-1})) ·
ι^{-1}
using *M.comp-assoc* **by** *auto*
also have ... = (*Comp a b d* · (*Comp b c d* ⊗ *Hom a b*)) ·
(a^{-1}[*Hom c d*, *Hom b c*, *Hom a b*] · (*h* ⊗ (*g* ⊗ *f*) · ι^{-1})) · ι^{-1}
using *a b c d A.Comp-assoc* **by** *auto*
also have ... = (*Comp a b d* · (*Comp b c d* ⊗ *Hom a b*)) ·
(a^{-1}[*Hom c d*, *Hom b c*, *Hom a b*] · (*h* ⊗ (*g* ⊗ *f*))) ·
(𝓘 ⊗ ι^{-1}) · ι^{-1}
proof −
have *h* ⊗ (*g* ⊗ *f*) · ι^{-1} = (*h* ⊗ (*g* ⊗ *f*)) · (𝓘 ⊗ ι^{-1})
proof −
have *M.seq h* 𝓘
using *h* **by** *auto*
moreover have *M.seq* (*g* ⊗ *f*) ι^{-1}
using *f g M.inv-in-hom M.unit-is-iso* **by** *blast*
ultimately show *?thesis*
using *a b c d f g h M.interchange M.comp-arr-ide M.ide-unity* **by** *metis*
qed
thus *?thesis*
using *M.comp-assoc* **by** *simp*
qed
also have ... = (*Comp a b d* · (*Comp b c d* ⊗ *Hom a b*)) ·
((*h* ⊗ *g*) ⊗ *f*) · a^{-1}[𝓘, 𝓘, 𝓘] · (𝓘 ⊗ ι^{-1}) · ι^{-1}
using *f g h M.assoc′-naturality*
by (*metis M.comp-assoc M.in-homE mem-Collect-eq*)
also have ... = (*Comp a b d* · (*Comp b c d* ⊗ *Hom a b*)) ·
(((*h* ⊗ *g*) ⊗ *f*) · (ι^{-1} ⊗ 𝓘)) · ι^{-1}
proof −
have a^{-1}[𝓘, 𝓘, 𝓘] · (𝓘 ⊗ ι^{-1}) · ι^{-1} = (ι^{-1} ⊗ 𝓘) · ι^{-1}
using *M.unitor-coincidence*
by (*metis* (*full-types*) *M.L.preserves-inv M.L.preserves-iso*
M.R.preserves-inv M.arrI M.arr-tensor M.comp-assoc M.ideD(*1*)
M.ide-unity M.inv-comp M.iso-assoc M.unit-in-hom-ax
M.unit-is-iso M.unit-triangle(*1*))
thus *?thesis*
using *M.comp-assoc* **by** *simp*
qed
also have ... = *Comp a b d* ·
((*Comp b c d* ⊗ *Hom a b*) · ((*h* ⊗ *g*) · ι^{-1} ⊗ *f*)) · ι^{-1}
proof −
have ((*h* ⊗ *g*) ⊗ *f*) · (ι^{-1} ⊗ 𝓘) = (*h* ⊗ *g*) · ι^{-1} ⊗ *f*
proof −
have *M.seq* (*h* ⊗ *g*) ι^{-1}
using *g h M.inv-in-hom M.unit-is-iso* **by** *blast*
moreover have *M.seq f* 𝓘
using *M.ide-in-hom M.ide-unity f* **by** *blast*

ultimately show *?thesis*
using *f g h M.interchange M.comp-arr-ide M.ide-unity* **by** *metis*
qed
thus *?thesis*
using *M.comp-assoc* **by** *auto*
qed
also have ... = $Comp\ a\ b\ d \cdot (Comp\ b\ c\ d \cdot (h \otimes g) \cdot \iota^{-1} \otimes f) \cdot \iota^{-1}$
using *b c d f g h 1 M.in-homE mem-Collect-eq M.comp-cod-arr M.interchange A.ide-Hom M.comp-ide-arr*
by *metis*
finally show $Comp\ a\ c\ d \cdot (h \otimes Comp\ a\ b\ c \cdot (g \otimes f) \cdot \iota^{-1}) \cdot \iota^{-1} = Comp\ a\ b\ d \cdot (Comp\ b\ c\ d \cdot (h \otimes g) \cdot \iota^{-1} \otimes f) \cdot \iota^{-1}$
by *blast*
qed
qed

abbreviation *comp* (**infixr** $\cdot_0$ *55*)
where $comp \equiv COMP$

lemma *hom-char*:
assumes $a \in Obj$ **and** $b \in Obj$
shows $hom\ (MkIde\ a)\ (MkIde\ b) = MkArr\ a\ b\ `\ M.hom\ \mathcal{I}\ (Hom\ a\ b)$
proof
show $hom\ (MkIde\ a)\ (MkIde\ b) \subseteq MkArr\ a\ b\ `\ M.hom\ \mathcal{I}\ (Hom\ a\ b)$
proof
fix *t*
assume *t*: $t \in hom\ (MkIde\ a)\ (MkIde\ b)$
have $t = MkArr\ a\ b\ (Map\ t)$
using *t MkArr-Map dom-char cod-char* **by** *fastforce*
moreover have $Map\ t \in M.hom\ \mathcal{I}\ (Hom\ a\ b)$
using *t arr-char dom-char cod-char* **by** *fastforce*
ultimately show $t \in MkArr\ a\ b\ `\ M.hom\ \mathcal{I}\ (Hom\ a\ b)$ **by** *simp*
qed
show $MkArr\ a\ b\ `\ M.hom\ \mathcal{I}\ (Hom\ a\ b) \subseteq hom\ (MkIde\ a)\ (MkIde\ b)$
using *assms MkArr-in-hom* **by** *blast*
qed

end

2.2.2 Underlying Functor

The underlying functor of an enriched functor $F : A \longrightarrow B$ takes an arrow «$f : a \to a'$» of the underlying category A_0 (*i.e.* an arrow «$\mathcal{I} \to Hom\ a\ a'$» of M) to the arrow «$F_a\ a\ a' \cdot f : F_o\ a \to F_o\ a'$» of B_0 (*i.e.* the arrow «$F_a\ a\ a' \cdot f : \mathcal{I} \to Hom\ (F_o\ a)\ (F_o\ a')$» of M).

locale *underlying-functor* =
enriched-functor
begin

sublocale A_0: *underlying-category C T α ι* Obj_A Hom_A Id_A $Comp_A$ **..**
sublocale B_0: *underlying-category C T α ι* Obj_B Hom_B Id_B $Comp_B$ **..**

notation A_0.*comp* (**infixr** $\cdot_{A0}$ *55*)
notation B_0.*comp* (**infixr** $\cdot_{B0}$ *55*)

definition map_0
where map_0 *f* = (*if* A_0.*arr f*
  then B_0.*MkArr* (F_o (A_0.*Dom f*)) (F_o (A_0.*Cod f*))
    (F_a (A_0.*Dom f*) (A_0.*Cod f*) · A_0.*Map f*)
  else B_0.*null*)

sublocale *functor* A_0.*comp* B_0.*comp* map_0
proof
 fix *f*
 show ¬ A_0.*arr f* ⟹ map_0 *f* = B_0.*null*
  using map_0-*def* **by** *simp*
 show *1*: ⋀*f*. A_0.*arr f* ⟹ B_0.*arr* (map_0 *f*)
 proof −
  fix *f*
  assume *f*: A_0.*arr f*
  have B_0.*arr* (B_0.*MkArr* (F_o (A_0.*Dom f*)) (F_o (A_0.*Cod f*))
     (F_a (A_0.*Dom f*) (A_0.*Cod f*) · A_0.*Map f*))
   using *f preserves-Hom* A_0.*Dom-in-Obj* A_0.*Cod-in-Obj* A_0.*arrE*
   by (*metis* (*mono-tags*, *lifting*) B_0.*arr-MkArr comp-in-homI*
    mem-Collect-eq preserves-Obj)
  thus B_0.*arr* (map_0 *f*)
   using *f* map_0-*def* **by** *simp*
 qed
 show A_0.*arr f* ⟹ B_0.*dom* (map_0 *f*) = map_0 (A_0.*dom f*)
  using *1* A_0.*dom-char* B_0.*dom-char preserves-Id* A_0.*arr-dom-iff-arr*
   map_0-*def* A_0.*Dom-in-Obj*
  by *auto*
 show A_0.*arr f* ⟹ B_0.*cod* (map_0 *f*) = map_0 (A_0.*cod f*)
  using *1* A_0.*cod-char* B_0.*cod-char preserves-Id* A_0.*arr-cod-iff-arr*
   map_0-*def* A_0.*Cod-in-Obj*
  by *auto*
 fix *g*
 assume *fg*: A_0.*seq g f*
 show map_0 ($g \cdot_{A0} f$) = map_0 $g \cdot_{B0}$ map_0 *f*
 proof −
  have B_0.*MkArr* (F_o (A_0.*Dom* ($g \cdot_{A0} f$))) (F_o (B_0.*Cod* ($g \cdot_{A0} f$)))
     (F_a (A_0.*Dom* ($g \cdot_{A0} f$))
     (B_0.*Cod* ($g \cdot_{A0} f$)) · B_0.*Map* ($g \cdot_{A0} f$)) =
    B_0.*MkArr* (F_o (A_0.*Dom g*)) (F_o (B_0.*Cod g*))
     (F_a (A_0.*Dom g*) (B_0.*Cod g*) · B_0.*Map g*) $\cdot_{B0}$
    B_0.*MkArr* (F_o (A_0.*Dom f*)) (F_o (B_0.*Cod f*))
     (F_a (A_0.*Dom f*) (B_0.*Cod f*) · B_0.*Map f*)
  proof −

have *2*: B_0*.arr* (B_0*.MkArr* (F_o (A_0*.Dom f*)) (F_o (A_0*.Dom g*))
(F_a (A_0*.Dom f*) (A_0*.Cod f*) · A_0*.Map f*))
using *fg 1* A_0*.seq-char* *map*$_0$*-def* **by** *auto*
have *3*: B_0*.arr* (B_0*.MkArr* (F_o (A_0*.Dom g*)) (F_o (A_0*.Cod g*))
(F_a (A_0*.Dom g*) (A_0*.Cod g*) · A_0*.Map g*))
using *fg 1* A_0*.seq-char* *map*$_0$*-def* **by** *metis*
have B_0*.MkArr* (F_o (A_0*.Dom g*)) (F_o (B_0*.Cod g*))
(F_a (A_0*.Dom g*) (B_0*.Cod g*) · B_0*.Map g*) $\cdot_{B0}$
B_0*.MkArr* (F_o (A_0*.Dom f*)) (F_o (B_0*.Cod f*))
(F_a (A_0*.Dom f*) (B_0*.Cod f*) · B_0*.Map f*) =
B_0*.MkArr* (F_o (A_0*.Dom f*)) (F_o (A_0*.Cod g*))
(*Comp*$_B$ (F_o (A_0*.Dom f*)) (F_o (A_0*.Dom g*)) (F_o (A_0*.Cod g*)) ·
(F_a (A_0*.Dom g*) (A_0*.Cod g*) · A_0*.Map g* ⊗
F_a (A_0*.Dom f*) (A_0*.Cod f*) · A_0*.Map f*) ·
ι^{-1})
using *fg 2 3* A_0*.seq-char* B_0*.comp-MkArr* **by** *simp*
moreover
have *Comp*$_B$ (F_o (A_0*.Dom f*)) (F_o (A_0*.Dom g*)) (F_o (A_0*.Cod g*)) ·
(F_a (A_0*.Dom g*) (A_0*.Cod g*) · A_0*.Map g* ⊗
F_a (A_0*.Dom f*) (A_0*.Cod f*) · A_0*.Map f*) · ι^{-1} =
F_a (A_0*.Dom* ($g \cdot_{A0} f$)) (B_0*.Cod* ($g \cdot_{A0} f$)) · B_0*.Map* ($g \cdot_{A0} f$)
proof –
have *Comp*$_B$ (F_o (A_0*.Dom f*)) (F_o (A_0*.Dom g*)) (F_o (A_0*.Cod g*)) ·
(F_a (A_0*.Dom g*) (A_0*.Cod g*) · A_0*.Map g* ⊗
F_a (A_0*.Dom f*) (A_0*.Cod f*) · A_0*.Map f*) · ι^{-1} =
Comp$_B$ (F_o (A_0*.Dom f*)) (F_o (A_0*.Dom g*)) (F_o (A_0*.Cod g*)) ·
((F_a (A_0*.Dom g*) (A_0*.Cod g*) ⊗ F_a (A_0*.Dom f*) (A_0*.Cod f*)) ·
(A_0*.Map g* ⊗ A_0*.Map f*)) · ι^{-1}
using *fg preserves-Hom*
interchange [*of* F_a (A_0*.Dom g*) (A_0*.Cod g*) A_0*.Map g*
F_a (A_0*.Dom f*) (A_0*.Cod f*) A_0*.Map f*]
by (*metis* A_0*.arrE* A_0*.seqE* *seqI′* *mem-Collect-eq*)
also have ... =
(*Comp*$_B$ (F_o (A_0*.Dom f*)) (F_o (A_0*.Dom g*)) (F_o (A_0*.Cod g*)) ·
(F_a (A_0*.Dom g*) (A_0*.Cod g*) ⊗ F_a (A_0*.Dom f*) (A_0*.Cod f*))) ·
(A_0*.Map g* ⊗ A_0*.Map f*) · ι^{-1}
using *comp-assoc* **by** *auto*
also have ... = (F_a (A_0*.Dom f*) (B_0*.Cod g*) ·
Comp$_A$ (A_0*.Dom f*) (A_0*.Dom g*) (B_0*.Cod g*)) ·
(A_0*.Map g* ⊗ A_0*.Map f*) · ι^{-1}
using *fg* A_0*.seq-char* *preserves-Comp* A_0*.Dom-in-Obj* A_0*.Cod-in-Obj*
by *auto*
also have ... = F_a (A_0*.Dom* ($g \cdot_{A0} f$)) (B_0*.Cod* ($g \cdot_{A0} f$)) ·
Comp$_A$ (A_0*.Dom f*) (A_0*.Dom g*) (B_0*.Cod g*) ·
(A_0*.Map g* ⊗ A_0*.Map f*) · ι^{-1}
using *fg comp-assoc* A_0*.seq-char* **by** *simp*
also have ... = F_a (A_0*.Dom* ($g \cdot_{A0} f$)) (B_0*.Cod* ($g \cdot_{A0} f$)) ·
B_0*.Map* ($g \cdot_{A0} f$)
using A_0*.Map-comp* A_0*.seq-char fg* **by** *presburger*

finally show *?thesis* **by** *blast*
qed
ultimately show *?thesis*
using A_0*.seq-char fg* **by** *auto*
qed
thus *?thesis*
using *fg* map_0*-def* B_0*.comp-MkArr* **by** *auto*
qed
qed

proposition *is-functor*:
shows *functor* A_0*.comp* B_0*.comp* map_0
..

end

2.2.3 Underlying Natural Transformation

The natural transformation underlying an enriched natural transformation τ has components that are essentially those of τ, except that we have to bother ourselves about coercions between types.

locale *underlying-natural-transformation* =
enriched-natural-transformation
begin

sublocale A_0: *underlying-category* $C\ T\ \alpha\ \iota\ Obj_A\ Hom_A\ Id_A\ Comp_A$ ..
sublocale B_0: *underlying-category* $C\ T\ \alpha\ \iota\ Obj_B\ Hom_B\ Id_B\ Comp_B$ ..
sublocale F_0: *underlying-functor* $C\ T\ \alpha\ \iota$
$Obj_A\ Hom_A\ Id_A\ Comp_A\ Obj_B\ Hom_B\ Id_B\ Comp_B\ F_o\ F_a$ ..
sublocale G_0: *underlying-functor* $C\ T\ \alpha\ \iota$
$Obj_A\ Hom_A\ Id_A\ Comp_A\ Obj_B\ Hom_B\ Id_B\ Comp_B\ G_o\ G_a$ ..

definition map_{obj}
where $map_{obj}\ a \equiv$
$B_0.MkArr\ (B_0.Dom\ (F_0.map_0\ a))\ (B_0.Dom\ (G_0.map_0\ a))$
$(\tau\ (A_0.Dom\ a))$

sublocale τ: *NaturalTransformation.transformation-by-components*
$A_0.comp\ B_0.comp\ F_0.map_0\ G_0.map_0\ map_{obj}$
proof
show $\bigwedge a.\ A_0.ide\ a \Longrightarrow B_0.in\text{-}hom\ (map_{obj}\ a)\ (F_0.map_0\ a)\ (G_0.map_0\ a)$
unfolding map_{obj}*-def*
using A_0*.Dom-in-Obj* $B_0.ide\text{-}char_{CC}$ $F_0.map_0$*-def* $G_0.map_0$*-def*
F_0*.preserves-ide* G_0*.preserves-ide* *component-in-hom*
by *auto*
show $\bigwedge f.\ A_0.arr\ f \Longrightarrow$
$map_{obj}\ (A_0.cod\ f) \cdot_{B0} F_0.map_0\ f =$
$G_0.map_0\ f \cdot_{B0} map_{obj}\ (A_0.dom\ f)$
proof −

fix f
assume f: $A_0.arr\ f$
show $map_{obj}\ (A_0.cod\ f) \cdot_{B0} F_0.map_0\ f =$
$G_0.map_0\ f \cdot_{B0} map_{obj}\ (A_0.dom\ f)$
proof (*intro* B_0.*arr-eqI*)
show *1*: $B_0.seq\ (map_{obj}\ (A_0.cod\ f))\ (F_0.map_0\ f)$
using A_0.*ide-cod*
‹$\bigwedge a.\ A_0.ide\ a \Longrightarrow$
$B_0.in\text{-}hom\ (map_{obj}\ a)\ (F_0.map_0\ a)\ (G_0.map_0\ a)$› f
by *blast*
show *2*: $B_0.seq\ (G_0.map_0\ f)\ (map_{obj}\ (A_0.dom\ f))$
using A_0.*ide-dom*
‹$\bigwedge a.\ A_0.ide\ a \Longrightarrow$
$B_0.in\text{-}hom\ (map_{obj}\ a)\ (F_0.map_0\ a)\ (G_0.map_0\ a)$› f
by *blast*
show $B_0.Dom\ (map_{obj}\ (A_0.cod\ f) \cdot_{B0} F_0.map_0\ f) =$
$B_0.Dom\ (G_0.map_0\ f \cdot_{B0} map_{obj}\ (A_0.dom\ f))$
using f *1 2* B_0.*comp-char* [*of* $map_{obj}\ (A_0.cod\ f)\ F_0.map_0\ f$]
B_0.*comp-char* [*of* $G_0.map_0\ f\ map_{obj}\ (A_0.dom\ f)$]
$F_0.map_0$-*def* $G_0.map_0$-*def* map_{obj}-*def*
by *simp*
show $B_0.Cod\ (map_{obj}\ (A_0.cod\ f) \cdot_{B0} F_0.map_0\ f) =$
$B_0.Cod\ (G_0.map_0\ f \cdot_{B0} map_{obj}\ (A_0.dom\ f))$
using f *1 2* B_0.*comp-char* [*of* $map_{obj}\ (A_0.cod\ f)\ F_0.map_0\ f$]
B_0.*comp-char* [*of* $G_0.map_0\ f\ map_{obj}\ (A_0.dom\ f)$]
$F_0.map_0$-*def* $G_0.map_0$-*def* map_{obj}-*def*
by *simp*
show $B_0.Map\ (map_{obj}\ (A_0.cod\ f) \cdot_{B0} F_0.map_0\ f) =$
$B_0.Map\ (G_0.map_0\ f \cdot_{B0} map_{obj}\ (A_0.dom\ f))$
proof −
have $Comp_B\ (F_o\ (A_0.Dom\ f))\ (F_o\ (A_0.Cod\ f))\ (G_o\ (A_0.Cod\ f)) \cdot$
$(\tau\ (A_0.Cod\ f) \otimes F_a\ (A_0.Dom\ f)\ (A_0.Cod\ f) \cdot A_0.Map\ f) \cdot \iota^{-1} =$
$Comp_B\ (F_o\ (A_0.Dom\ f))\ (G_o\ (A_0.Dom\ f))\ (G_o\ (A_0.Cod\ f)) \cdot$
$(G_a\ (A_0.Dom\ f)\ (A_0.Cod\ f) \cdot A_0.Map\ f \otimes \tau\ (A_0.Dom\ f)) \cdot \iota^{-1}$
proof −
have $Comp_B\ (F_o\ (A_0.Dom\ f))\ (F_o\ (A_0.Cod\ f))\ (G_o\ (A_0.Cod\ f)) \cdot$
$(\tau\ (A_0.Cod\ f) \otimes F_a\ (A_0.Dom\ f)\ (A_0.Cod\ f) \cdot A_0.Map\ f) \cdot \iota^{-1} =$
$Comp_B\ (F_o\ (A_0.Dom\ f))\ (F_o\ (A_0.Cod\ f))\ (G_o\ (A_0.Cod\ f)) \cdot$
$((\tau\ (A_0.Cod\ f) \otimes F_a\ (A_0.Dom\ f)\ (A_0.Cod\ f)) \cdot (\mathcal{I} \otimes A_0.Map\ f)) \cdot$
ι^{-1}
proof −
have $\tau\ (A_0.Cod\ f) \otimes F_a\ (A_0.Dom\ f)\ (A_0.Cod\ f) \cdot A_0.Map\ f =$
$(\tau\ (A_0.Cod\ f) \otimes F_a\ (A_0.Dom\ f)\ (A_0.Cod\ f)) \cdot (\mathcal{I} \otimes A_0.Map\ f)$
proof −
have $seq\ (\tau\ (A_0.Cod\ f))\ \mathcal{I}$
using f *seqI component-in-hom*
by (*metis* (*no-types, lifting*) A_0.*Cod-in-Obj ide-char*
ide-unity in-homE)
moreover have $seq\ (F_a\ (A_0.Dom\ f)\ (B_0.Cod\ f))\ (B_0.Map\ f)$

using f A_0.*Map-in-Hom* A_0.*Cod-in-Obj* A_0.*Dom-in-Obj*
F.preserves-Hom in-homE
by *blast*
ultimately show *?thesis*
using *f component-in-hom interchange comp-arr-dom* **by** *auto*
qed
thus *?thesis* **by** *simp*
qed
also have ... =
$Comp_B\ (F_o\ (A_0.Dom\ f))\ (F_o\ (A_0.Cod\ f))\ (G_o\ (A_0.Cod\ f))\ \cdot$
$((\tau\ (B_0.Cod\ f) \otimes F_a\ (A_0.Dom\ f)\ (B_0.Cod\ f))\ \cdot$
$(\mathrm{l}^{-1}[Hom_A\ (A_0.Dom\ f)\ (B_0.Cod\ f)]\ \cdot$
$\mathrm{l}[Hom_A\ (A_0.Dom\ f)\ (B_0.Cod\ f)])\ \cdot$
$(\mathcal{I} \otimes B_0.Map\ f))\ \cdot\ \iota^{-1}$
proof −
have $(\mathrm{l}^{-1}[Hom_A\ (A_0.Dom\ f)\ (B_0.Cod\ f)]\ \cdot$
$\mathrm{l}[Hom_A\ (A_0.Dom\ f)\ (B_0.Cod\ f)])\ \cdot$
$(\mathcal{I} \otimes B_0.Map\ f) =$
$\mathcal{I} \otimes B_0.Map\ f$
using *f comp-lunit-lunit'(2)*
by (*metis* (*no-types*, *lifting*) *A.ide-Hom* A_0.*arrE comp-cod-arr*
comp-ide-self ideD(1) ide-unity interchange in-homE
mem-Collect-eq)
thus *?thesis* **by** *simp*
qed
also have ... =
$(Comp_B\ (F_o\ (A_0.Dom\ f))\ (F_o\ (B_0.Cod\ f))\ (G_o\ (B_0.Cod\ f))\ \cdot$
$(\tau\ (B_0.Cod\ f) \otimes F_a\ (A_0.Dom\ f)\ (B_0.Cod\ f))\ \cdot$
$\mathrm{l}^{-1}[Hom_A\ (A_0.Dom\ f)\ (B_0.Cod\ f)])\ \cdot$
$\mathrm{l}[Hom_A\ (A_0.Dom\ f)\ (B_0.Cod\ f)]\ \cdot\ (\mathcal{I} \otimes B_0.Map\ f)\ \cdot\ \iota^{-1}$
using *comp-assoc* **by** *simp*
also have ... =
$Comp_B\ (F_o\ (A_0.Dom\ f))\ (G_o\ (A_0.Dom\ f))\ (G_o\ (B_0.Cod\ f))\ \cdot$
$(G_a\ (A_0.Dom\ f)\ (B_0.Cod\ f) \otimes \tau\ (A_0.Dom\ f))\ \cdot$
$\mathrm{r}^{-1}[Hom_A\ (A_0.Dom\ f)\ (B_0.Cod\ f)]\ \cdot$
$(\mathrm{l}[Hom_A\ (A_0.Dom\ f)\ (B_0.Cod\ f)]\ \cdot\ (\mathcal{I} \otimes B_0.Map\ f))\ \cdot\ \iota^{-1}$
using f A_0.*Cod-in-Obj* A_0.*Dom-in-Obj naturality comp-assoc* **by** *simp*
also have ... =
$Comp_B\ (F_o\ (A_0.Dom\ f))\ (G_o\ (A_0.Dom\ f))\ (G_o\ (B_0.Cod\ f))\ \cdot$
$(G_a\ (A_0.Dom\ f)\ (B_0.Cod\ f) \otimes \tau\ (A_0.Dom\ f))\ \cdot$
$\mathrm{r}^{-1}[Hom_A\ (A_0.Dom\ f)\ (B_0.Cod\ f)]\ \cdot\ (B_0.Map\ f\ \cdot\ \mathrm{l}[\mathcal{I}])\ \cdot\ \iota^{-1}$
using *f lunit-naturality* A_0.*Map-in-Hom* **by** *force*
also have ... =
$Comp_B\ (F_o\ (A_0.Dom\ f))\ (G_o\ (A_0.Dom\ f))\ (G_o\ (B_0.Cod\ f))\ \cdot$
$(G_a\ (A_0.Dom\ f)\ (B_0.Cod\ f) \otimes \tau\ (A_0.Dom\ f))\ \cdot$
$\mathrm{r}^{-1}[Hom_A\ (A_0.Dom\ f)\ (B_0.Cod\ f)]\ \cdot\ B_0.Map\ f$
proof −
have $\iota\ \cdot\ \iota^{-1} = \mathcal{I}$
using *comp-arr-inv' unit-is-iso* **by** *blast*

moreover have «$B_0.Map\ f : \mathcal{I} \rightarrow Hom_A\ (A_0.Dom\ f)\ (B_0.Cod\ f)$»
using *f* A_0*.Map-in-Hom* **by** *blast*
ultimately show *?thesis*
using *f comp-arr-dom unitor-coincidence*(*1*) *comp-assoc* **by** *auto*
qed
also have ... =
$Comp_B\ (F_o\ (A_0.Dom\ f))\ (G_o\ (A_0.Dom\ f))\ (G_o\ (B_0.Cod\ f)) \cdot$
$(G_a\ (A_0.Dom\ f)\ (B_0.Cod\ f) \otimes \tau\ (A_0.Dom\ f)) \cdot$
$(B_0.Map\ f \otimes \mathcal{I}) \cdot \mathrm{r}^{-1}[\mathcal{I}]$
using *f runit′-naturality* A_0*.Map-in-Hom* **by** *force*
also have ... =
$Comp_B\ (F_o\ (A_0.Dom\ f))\ (G_o\ (A_0.Dom\ f))\ (G_o\ (B_0.Cod\ f)) \cdot$
$((G_a\ (A_0.Dom\ f)\ (B_0.Cod\ f) \otimes \tau\ (A_0.Dom\ f)) \cdot$
$(B_0.Map\ f \otimes \mathcal{I})) \cdot \iota^{-1}$
using *unitor-coincidence comp-assoc* **by** *simp*
also have ... =
$Comp_B\ (F_o\ (A_0.Dom\ f))\ (G_o\ (A_0.Dom\ f))\ (G_o\ (B_0.Cod\ f)) \cdot$
$(G_a\ (A_0.Dom\ f)\ (B_0.Cod\ f) \cdot A_0.Map\ f \otimes \tau\ (A_0.Dom\ f)) \cdot \iota^{-1}$
proof −
have $seq\ (G_a\ (A_0.Dom\ f)\ (B_0.Cod\ f))\ (B_0.Map\ f)$
using *f* A_0*.Map-in-Hom* A_0*.Cod-in-Obj* A_0*.Dom-in-Obj G.preserves-Hom*
by *fast*
moreover have $seq\ (\tau\ (A_0.Dom\ f))\ \mathcal{I}$
using *f seqI component-in-hom*
by (*metis* (*no-types*, *lifting*) A_0*.Dom-in-Obj ide-char*
ide-unity in-homE)
ultimately show *?thesis*
using *f comp-arr-dom interchange* **by** *auto*
qed
finally show *?thesis* **by** *simp*
qed
thus *?thesis*
using *f 1 2* B_0*.comp-char* [*of* $map_{obj}\ (A_0.cod\ f)\ F_0.map_0\ f$]
B_0*.comp-char* [*of* $G_0.map_0\ f\ map_{obj}\ (A_0.dom\ f)$]
$F_0.map_0$*-def* $G_0.map_0$*-def* map_{obj}*-def*
by *simp*
qed
qed
qed
qed

proposition *is-natural-transformation*:
shows *natural-transformation* $A_0.comp\ B_0.comp\ F_0.map_0\ G_0.map_0\ \tau.map$
..

end

2.2.4 Self-Enriched Case

Here we show that a closed monoidal category C, regarded as a category enriched in itself, it is isomorphic to its own underlying category. This is useful, because it is somewhat less cumbersome to work directly in the category C than in the higher-type version that results from the underlying category construction. Kelly often regards these two categories as identical.

locale *self-enriched-category* =
  elementary-closed-monoidal-category +
  enriched-category C T α ι ‹Collect ide› exp Id Comp
begin

  sublocale *UC*: *underlying-category C T α ι ‹Collect ide› exp Id Comp* **..**

  abbreviation *toUC*
  where *toUC g* ≡ *if arr g*
        then UC.MkArr (dom g) (cod g) $(g^{\uparrow})$
        else UC.null

  lemma *toUC-simps* [*simp*]:
  assumes *arr f*
  shows *UC.arr (toUC f)*
  and *UC.dom (toUC f) = toUC (dom f)*
  and *UC.cod (toUC f) = toUC (cod f)*
    using *assms UC.arr-char UC.dom-char UC.cod-char UP-def*
      comp-cod-arr Id-def
    by *auto*

  lemma *toUC-in-hom* [*intro*]:
  assumes *arr f*
  shows *UC.in-hom (toUC f) (UC.MkIde (dom f)) (UC.MkIde (cod f))*
    using *assms toUC-simps* **by** *fastforce*

  sublocale *toUC*: *functor C UC.comp toUC*
      using *toUC-simps UP-comp UC.COMP-def*
      by *unfold-locales auto*

  abbreviation *frmUC*
  where *frmUC g* ≡ *if UC.arr g*
        then $(UC.Map\ g)^{\downarrow}[UC.Dom\ g,\ UC.Cod\ g]$
        else null

  lemma *frmUC-simps* [*simp*]:
  assumes *UC.arr f*
  shows *arr (frmUC f)*
  and *dom (frmUC f) = frmUC (UC.dom f)*
  and *cod (frmUC f) = frmUC (UC.cod f)*
    using *assms UC.arr-char UC.dom-char UC.cod-char Uncurry-Curry*
      Id-def lunit-in-hom DN-def

by *auto*

lemma *frmUC-in-hom* [*intro*]:
assumes *UC.in-hom f a b*
shows «*frmUC f* : *frmUC a* → *frmUC b*»
using *assms frmUC-simps* **by** *blast*

lemma *DN-Map-comp*:
assumes *UC.seq g f*
shows $(UC.Map\ (UC.comp\ g\ f))^{\downarrow}[UC.Dom\ f,\ UC.Cod\ g] =$
$(UC.Map\ g)^{\downarrow}[UC.Dom\ g,\ UC.Cod\ g] \cdot$
$(UC.Map\ f)^{\downarrow}[UC.Dom\ f,\ UC.Cod\ f]$
proof −
have $(UC.Map\ (UC.comp\ g\ f))^{\downarrow}[UC.Dom\ f,\ UC.Cod\ g] =$
$((UC.Map\ (UC.comp\ g\ f))^{\downarrow}[UC.Dom\ f,\ UC.Cod\ g])^{\uparrow}$
${}^{\downarrow}[UC.Dom\ f,\ UC.Cod\ g]$
using *assms UC.arr-char UC.seq-char* [*of g f*] **by** *fastforce*
also have ... $= ((UC.Map\ g)^{\downarrow}[UC.Dom\ g,\ UC.Cod\ g] \cdot$
$(UC.Map\ f)^{\downarrow}[UC.Dom\ f,\ UC.Cod\ f])^{\uparrow}$
${}^{\downarrow}[UC.Dom\ f,\ UC.Cod\ g]$
proof −
have $((UC.Map\ (UC.comp\ g\ f))^{\downarrow}[UC.Dom\ f,\ UC.Cod\ g])^{\uparrow} =$
$UC.Map\ (UC.comp\ g\ f)$
using *assms UC.arr-char UC.seq-char* [*of g f*] **by** *fastforce*
also have ... $= Comp\ (UC.Dom\ f)\ (UC.Dom\ g)\ (UC.Cod\ g) \cdot$
$(UC.Map\ g \otimes UC.Map\ f) \cdot \iota^{-1}$
using *assms UC.Map-comp UC.seq-char* **by** *blast*
also have ... $= Comp\ (UC.Dom\ f)\ (UC.Dom\ g)\ (UC.Cod\ g) \cdot$
$(((UC.Map\ g)^{\downarrow}[UC.Dom\ g,\ UC.Cod\ g])^{\uparrow} \otimes$
$((UC.Map\ f)^{\downarrow}[UC.Dom\ f,\ UC.Cod\ f])^{\uparrow}) \cdot \iota^{-1}$
using *assms UC.seq-char UC.arr-char* **by** *auto*
also have ... $= ((UC.Map\ g)^{\downarrow}[UC.Dom\ g,\ UC.Cod\ g] \cdot$
$(UC.Map\ f)^{\downarrow}[UC.Dom\ f,\ UC.Cod\ f])^{\uparrow}$
proof −
have $dom\ ((UC.Map\ f)^{\downarrow}[UC.Dom\ f,\ UC.Cod\ f]) = UC.Dom\ f$
using *assms DN-Id UC.Dom-in-Obj frmUC-simps*(*2*) **by** *auto*
moreover have $cod\ ((UC.Map\ f)^{\downarrow}[UC.Dom\ f,\ UC.Cod\ f]) = UC.Cod\ f$
using *assms DN-Id UC.Cod-in-Obj frmUC-simps*(*3*) **by** *auto*
moreover have $seq\ ((UC.Map\ g)^{\downarrow}[UC.Cod\ f,\ UC.Cod\ g])$
$((UC.Map\ f)^{\downarrow}[UC.Dom\ f,\ UC.Cod\ f])$
using *assms frmUC-simps*(*1*−*3*) *UC.seq-char*
apply (*intro seqI*)
apply *auto*[*3*]
by *metis*+
ultimately show *?thesis*
using *assms UP-comp UP-DN*(*2*) *UC.arr-char UC.seq-char*
in-homE seqI
by *auto*
qed

finally show *?thesis* **by** *simp*
qed
also have ... = $(UC.Map\ g)^{\downarrow}[UC.Dom\ g,\ UC.Cod\ g]$ ·
$(UC.Map\ f)^{\downarrow}[UC.Dom\ f,\ UC.Cod\ f]$
proof −
have *2*: *seq* $((UC.Map\ g)^{\downarrow}[UC.Dom\ g,\ UC.Cod\ g])$
$((UC.Map\ f)^{\downarrow}[UC.Dom\ f,\ UC.Cod\ f])$
using *assms frmUC-simps*(*1*−*3*) *UC.seq-char*
apply (*elim UC.seqE*, *intro seqI*)
apply *auto*[*3*]
by *metis*+
moreover have *dom* $((UC.Map\ g)^{\downarrow}[UC.Dom\ g,\ UC.Cod\ g]$ ·
$(UC.Map\ f)^{\downarrow}[UC.Dom\ f,\ UC.Cod\ f])$ =
UC.Dom f
using *assms 2 UC.Dom-comp UC.arr-char* [*of f*] **by** *auto*
moreover have *cod* $((UC.Map\ g)^{\downarrow}[UC.Dom\ g,\ UC.Cod\ g]$ ·
$(UC.Map\ f)^{\downarrow}[UC.Dom\ f,\ UC.Cod\ f])$ =
UC.Cod g
using *assms 2 UC.Cod-comp UC.arr-char* [*of g*] **by** *auto*
ultimately show *?thesis*
using *assms*
UP-DN(*3*) [*of* $(UC.Map\ g)^{\downarrow}[UC.Dom\ g,\ UC.Cod\ g]$ ·
$(UC.Map\ f)^{\downarrow}[UC.Dom\ f,\ UC.Cod\ f]$]
by *simp*
qed
finally show *?thesis* **by** *blast*
qed

sublocale *frmUC*: *functor UC.comp C frmUC*
proof
show ⋀*f*. ¬ *UC.arr f* ⟹ *frmUC f* = *null*
by *simp*
show ⋀*f*. *UC.arr f* ⟹ *arr* (*frmUC f*)
using *UC.arr-char frmUC-simps*(*1*) **by** *blast*
show ⋀*f*. *UC.arr f* ⟹ *dom* (*frmUC f*) = *frmUC* (*UC.dom f*)
using *frmUC-simps*(*2*) **by** *blast*
show ⋀*f*. *UC.arr f* ⟹ *cod* (*frmUC f*) = *frmUC* (*UC.cod f*)
using *frmUC-simps*(*3*) **by** *blast*
fix *f g*
assume *fg*: *UC.seq g f*
show *frmUC* (*UC.comp g f*) = *frmUC g* · *frmUC f*
using *fg UC.seq-char DN-Map-comp* **by** *auto*
qed

sublocale *inverse-functors UC.comp C toUC frmUC*
proof
show *frmUC* ∘ *toUC* = *map*
using *is-extensional comp-arr-dom comp-assoc Uncurry-Curry* **by** *auto*
interpret *to-frm*: *composite-functor UC.comp C UC.comp frmUC toUC* **..**

```
show toUC ∘ frmUC = UC.map
proof
  fix f
  show (toUC ∘ frmUC) f = UC.map f
  proof (cases UC.arr f)
    show ¬ UC.arr f ⟹ ?thesis
      using UC.is-extensional by auto
    assume f: UC.arr f
    show ?thesis
    proof (intro UC.arr-eqI)
      show UC.arr ((toUC ∘ frmUC) f)
        using f by blast
      show UC.arr (UC.map f)
        using f by blast
      show UC.Dom ((toUC ∘ frmUC) f) = UC.Dom (UC.map f)
        using f UC.Dom-in-Obj frmUC.preserves-arr UC.arr-char [of f]
        by auto
      show UC.Cod (to-frm.map f) = UC.Cod (UC.map f)
        using f UC.arr-char [of f] by auto
      show UC.Map (to-frm.map f) = UC.Map (UC.map f)
        using f UP-DN UC.arr-char [of f] by auto
    qed
  qed
qed
qed

lemma inverse-functors-toUC-frmUC:
shows inverse-functors UC.comp C toUC frmUC
  ..

corollary enriched-category-isomorphic-to-underlying-category:
shows isomorphic-categories UC.comp C
  using inverse-functors-toUC-frmUC
  by unfold-locales blast

end
```

2.3 Opposite of an Enriched Category

Construction of the opposite of an enriched category (*cf.* [1] (1.19)) requires that the underlying monoidal category be symmetric, in order to introduce the required "twist" in the definition of composition.

```
locale opposite-enriched-category =
  symmetric-monoidal-category +
  EC: enriched-category
begin

  interpretation elementary-symmetric-monoidal-category
```

C tensor unity lunit runit assoc sym
using *induces-elementary-symmetric-monoidal-category*$_{CMC}$ **by** *blast*

abbreviation (*input*) Hom_{op}
where Hom_{op} *a b* ≡ *Hom b a*

abbreviation $Comp_{op}$
where $Comp_{op}$ *a b c* ≡ *Comp c b a* · s[*Hom c b*, *Hom b a*]

sublocale *enriched-category C T α ι Obj* Hom_{op} *Id* $Comp_{op}$
proof
show ∗: ⋀*a b*. ⟦*a* ∈ *Obj*; *b* ∈ *Obj*⟧ ⟹ *ide* (*Hom b a*)
using *EC.ide-Hom* **by** *blast*
show ⋀*a*. *a* ∈ *Obj* ⟹ «*Id a* : *ℐ* → *Hom a a*»
using *EC.Id-in-hom* **by** *blast*
show ∗∗: ⋀*a b c*. ⟦*a* ∈ *Obj*; *b* ∈ *Obj*; *c* ∈ *Obj*⟧ ⟹
«$Comp_{op}$ *a b c* : *Hom c b* ⊗ *Hom b a* → *Hom c a*»
using *sym-in-hom EC.ide-Hom EC.Comp-in-hom* **by** *auto*
show ⋀*a b*. ⟦*a* ∈ *Obj*; *b* ∈ *Obj*⟧ ⟹
$Comp_{op}$ *a a b* · (*Hom b a* ⊗ *Id a*) = r[*Hom b a*]
proof −
fix *a b*
assume *a*: *a* ∈ *Obj* **and** *b*: *b* ∈ *Obj*
have $Comp_{op}$ *a a b* · (*Hom b a* ⊗ *Id a*) =
Comp b a a · s[*Hom b a*, *Hom a a*] · (*Hom b a* ⊗ *Id a*)
using *comp-assoc* **by** *simp*
also have ... = *Comp b a a* · (*Id a* ⊗ *Hom b a*) · s[*Hom b a*, *ℐ*]
using *a b sym-naturality* [*of Hom b a Id a*] *sym-in-hom*
EC.Id-in-hom EC.ide-Hom
by *fastforce*
also have ... = (*Comp b a a* · (*Id a* ⊗ *Hom b a*)) · s[*Hom b a*, *ℐ*]
using *comp-assoc* **by** *simp*
also have ... = l[*Hom b a*] · s[*Hom b a*, *ℐ*]
using *a b EC.Comp-Id-Hom* **by** *simp*
also have ... = r[*Hom b a*]
using *a b unitor-coherence EC.ide-Hom* **by** *presburger*
finally show $Comp_{op}$ *a a b* · (*Hom b a* ⊗ *Id a*) = r[*Hom b a*]
by *blast*
qed
show ⋀*a b*. ⟦*a* ∈ *Obj*; *b* ∈ *Obj*⟧ ⟹
$Comp_{op}$ *a b b* · (*Id b* ⊗ *Hom b a*) = l[*Hom b a*]
proof −
fix *a b*
assume *a*: *a* ∈ *Obj* **and** *b*: *b* ∈ *Obj*
have $Comp_{op}$ *a b b* · (*Id b* ⊗ *Hom b a*) =
Comp b b a · s[*Hom b b*, *Hom b a*] · (*Id b* ⊗ *Hom b a*)
using *comp-assoc* **by** *simp*
also have ... = *Comp b b a* · (*Hom b a* ⊗ *Id b*) · s[*ℐ*, *Hom b a*]

using a b *sym-naturality* [*of Id b Hom b a*] *sym-in-hom*
EC.Id-in-hom EC.ide-Hom
by *force*
also have ... = $(Comp\ b\ b\ a \cdot (Hom\ b\ a \otimes Id\ b)) \cdot \mathrm{s}[\mathcal{I}, Hom\ b\ a]$
using *comp-assoc* **by** *simp*
also have ... = $\mathrm{r}[Hom\ b\ a] \cdot \mathrm{s}[\mathcal{I}, Hom\ b\ a]$
using a b *EC.Comp-Hom-Id* **by** *simp*
also have ... = $\mathrm{l}[Hom\ b\ a]$
proof −

have $\mathrm{r}[Hom\ b\ a] \cdot \mathrm{s}[\mathcal{I}, Hom\ b\ a] =$
$(\mathrm{l}[Hom\ b\ a] \cdot \mathrm{s}[Hom\ b\ a, \mathcal{I}]) \cdot \mathrm{s}[\mathcal{I}, Hom\ b\ a]$
using a b *unitor-coherence EC.ide-Hom* **by** *simp*
also have ... = $\mathrm{l}[Hom\ b\ a] \cdot \mathrm{s}[Hom\ b\ a, \mathcal{I}] \cdot \mathrm{s}[\mathcal{I}, Hom\ b\ a]$
using *comp-assoc* **by** *simp*
also have ... = $\mathrm{l}[Hom\ b\ a]$
using a b *comp-arr-dom comp-arr-inv sym-inverse* **by** *simp*
finally show *?thesis* **by** *blast*
qed
finally show $Comp_{op}\ a\ b\ b \cdot (Id\ b \otimes Hom\ b\ a) = \mathrm{l}[Hom\ b\ a]$
by *blast*
qed
show $\bigwedge a\ b\ c\ d.\ [\![a \in Obj;\ b \in Obj;\ c \in Obj;\ d \in Obj]\!] \Longrightarrow$
$Comp_{op}\ a\ b\ d \cdot (Comp_{op}\ b\ c\ d \otimes Hom\ b\ a) =$
$Comp_{op}\ a\ c\ d \cdot (Hom\ d\ c \otimes Comp_{op}\ a\ b\ c) \cdot$
$\mathrm{a}[Hom\ d\ c, Hom\ c\ b, Hom\ b\ a]$
proof −
fix $a\ b\ c\ d$
assume a: $a \in Obj$ **and** b: $b \in Obj$ **and** c: $c \in Obj$ **and** d: $d \in Obj$
have $Comp_{op}\ a\ b\ d \cdot (Comp_{op}\ b\ c\ d \otimes Hom\ b\ a) =$
$Comp_{op}\ a\ b\ d \cdot (Comp\ d\ c\ b \otimes Hom\ b\ a) \cdot$
$(\mathrm{s}[Hom\ d\ c, Hom\ c\ b] \otimes Hom\ b\ a)$
using a b c d ∗∗ *interchange comp-ide-arr ide-in-hom seqI′*
EC.ide-Hom
by *metis*
also have ... = $(Comp\ d\ b\ a \cdot$
$(\mathrm{s}[Hom\ d\ b, Hom\ b\ a] \cdot (Comp\ d\ c\ b \otimes Hom\ b\ a)) \cdot$
$(\mathrm{s}[Hom\ d\ c, Hom\ c\ b] \otimes Hom\ b\ a))$
using *comp-assoc* **by** *simp*
also have ... = $(Comp\ d\ b\ a \cdot$
$((Hom\ b\ a \otimes Comp\ d\ c\ b) \cdot$
$\mathrm{s}[Hom\ c\ b \otimes Hom\ d\ c, Hom\ b\ a]) \cdot$
$(\mathrm{s}[Hom\ d\ c, Hom\ c\ b] \otimes Hom\ b\ a))$
using a b c d *sym-naturality EC.Comp-in-hom ide-char*
in-homE EC.ide-Hom
by *metis*
also have ... = $(Comp\ d\ b\ a \cdot (Hom\ b\ a \otimes Comp\ d\ c\ b)) \cdot$
$(\mathrm{s}[Hom\ c\ b \otimes Hom\ d\ c, Hom\ b\ a] \cdot$
$(\mathrm{s}[Hom\ d\ c, Hom\ c\ b] \otimes Hom\ b\ a))$

using *comp-assoc* **by** *simp*
also have ... = (*Comp d c a* · (*Comp c b a* ⊗ *Hom d c*) ·
a^{-1}[*Hom b a*, *Hom c b*, *Hom d c*]) ·
(s[*Hom c b* ⊗ *Hom d c*, *Hom b a*] ·
(s[*Hom d c*, *Hom c b*] ⊗ *Hom b a*))
proof −
have *Comp d b a* · (*Hom b a* ⊗ *Comp d c b*) =
(*Comp d b a* · (*Hom b a* ⊗ *Comp d c b*)) ·
(*Hom b a* ⊗ *Hom c b* ⊗ *Hom d c*)
using *a b c d EC.Comp-in-hom arrI comp-in-homI ide-in-hom*
tensor-in-hom EC.ide-Hom
proof −
have *seq* (*Comp d b a*) (*Hom b a* ⊗ *Comp d c b*)
using *a b c d EC.Comp-in-hom arrI comp-in-homI ide-in-hom*
tensor-in-hom EC.ide-Hom
by *meson*
moreover have *dom* (*Comp d b a* · (*Hom b a* ⊗ *Comp d c b*)) =
(*Hom b a* ⊗ *Hom c b* ⊗ *Hom d c*)
using *a b c d EC.Comp-in-hom dom-comp dom-tensor ideD*(*1*−*2*)
in-homE calculation EC.ide-Hom
by *metis*
ultimately show *?thesis*
using *a b c d EC.Comp-in-hom comp-arr-dom* **by** *metis*
qed
also have ... =
(*Comp d b a* · (*Hom b a* ⊗ *Comp d c b*)) ·
a[*Hom b a*, *Hom c b*, *Hom d c*] · a^{-1}[*Hom b a*, *Hom c b*, *Hom d c*]
using *a b c d comp-assoc-assoc′*(*1*) *EC.ide-Hom* **by** *simp*
also have ... = (*Comp d b a* · (*Hom b a* ⊗ *Comp d c b*) ·
a[*Hom b a*, *Hom c b*, *Hom d c*]) ·
a^{-1}[*Hom b a*, *Hom c b*, *Hom d c*]
using *comp-assoc* **by** *simp*
also have ... = (*Comp d c a* · (*Comp c b a* ⊗ *Hom d c*)) ·
a^{-1}[*Hom b a*, *Hom c b*, *Hom d c*]
using *a b c d EC.Comp-assoc* **by** *simp*
also have ... = *Comp d c a* · (*Comp c b a* ⊗ *Hom d c*) ·
a^{-1}[*Hom b a*, *Hom c b*, *Hom d c*]
using *comp-assoc* **by** *simp*
finally have *Comp d b a* · (*Hom b a* ⊗ *Comp d c b*) =
Comp d c a · (*Comp c b a* ⊗ *Hom d c*) ·
a^{-1}[*Hom b a*, *Hom c b*, *Hom d c*]
by *blast*
thus *?thesis* **by** *simp*
qed
also have ... = (*Comp d c a* · (*Comp c b a* ⊗ *Hom d c*)) ·
(a^{-1}[*Hom b a*, *Hom c b*, *Hom d c*] ·
s[*Hom c b* ⊗ *Hom d c*, *Hom b a*] ·
(s[*Hom d c*, *Hom c b*] ⊗ *Hom b a*))
using *comp-assoc* **by** *simp*

finally have *LHS*: $(Comp\ d\ b\ a \cdot \mathrm{s}[Hom\ d\ b,\ Hom\ b\ a]) \cdot$
$(Comp\ d\ c\ b \cdot \mathrm{s}[Hom\ d\ c,\ Hom\ c\ b] \otimes Hom\ b\ a) =$
$(Comp\ d\ c\ a \cdot (Comp\ c\ b\ a \otimes Hom\ d\ c)) \cdot$
$(\mathrm{a}^{-1}[Hom\ b\ a,\ Hom\ c\ b,\ Hom\ d\ c] \cdot$
$\mathrm{s}[Hom\ c\ b \otimes Hom\ d\ c,\ Hom\ b\ a] \cdot$
$(\mathrm{s}[Hom\ d\ c,\ Hom\ c\ b] \otimes Hom\ b\ a))$
by *blast*
have $Comp_{op}\ a\ c\ d \cdot (Hom\ d\ c \otimes Comp_{op}\ a\ b\ c) \cdot$
$\mathrm{a}[Hom\ d\ c,\ Hom\ c\ b,\ Hom\ b\ a] =$
$Comp\ d\ c\ a \cdot$
$(\mathrm{s}[Hom\ d\ c,\ Hom\ c\ a] \cdot$
$(Hom\ d\ c \otimes Comp\ c\ b\ a \cdot \mathrm{s}[Hom\ c\ b,\ Hom\ b\ a])) \cdot$
$\mathrm{a}[Hom\ d\ c,\ Hom\ c\ b,\ Hom\ b\ a]$
using *comp-assoc* **by** *simp*
also have ... =
$Comp\ d\ c\ a \cdot$
$((Comp\ c\ b\ a \cdot \mathrm{s}[Hom\ c\ b,\ Hom\ b\ a] \otimes Hom\ d\ c) \cdot$
$\mathrm{s}[Hom\ d\ c,\ Hom\ c\ b \otimes Hom\ b\ a]) \cdot$
$\mathrm{a}[Hom\ d\ c,\ Hom\ c\ b,\ Hom\ b\ a]$
using *a b c d* ∗∗ *sym-naturality ide-char in-homE EC.ide-Hom*
by *metis*
also have ... =
$Comp\ d\ c\ a \cdot$
$(((Comp\ c\ b\ a \otimes Hom\ d\ c) \cdot (\mathrm{s}[Hom\ c\ b,\ Hom\ b\ a] \otimes Hom\ d\ c)) \cdot$
$\mathrm{s}[Hom\ d\ c,\ Hom\ c\ b \otimes Hom\ b\ a]) \cdot$
$\mathrm{a}[Hom\ d\ c,\ Hom\ c\ b,\ Hom\ b\ a]$
using *a b c d* ∗∗ *interchange comp-arr-dom ideD(1−2)*
in-homE EC.ide-Hom
by *metis*
also have ... = $(Comp\ d\ c\ a \cdot (Comp\ c\ b\ a \otimes Hom\ d\ c)) \cdot$
$((\mathrm{s}[Hom\ c\ b,\ Hom\ b\ a] \otimes Hom\ d\ c) \cdot$
$\mathrm{s}[Hom\ d\ c,\ Hom\ c\ b \otimes Hom\ b\ a] \cdot$
$\mathrm{a}[Hom\ d\ c,\ Hom\ c\ b,\ Hom\ b\ a])$
using *comp-assoc* **by** *simp*
also have ... = $(Comp\ d\ c\ a \cdot (Comp\ c\ b\ a \otimes Hom\ d\ c)) \cdot$
$(\mathrm{a}^{-1}[Hom\ b\ a,\ Hom\ c\ b,\ Hom\ d\ c] \cdot$
$\mathrm{s}[Hom\ c\ b \otimes Hom\ d\ c,\ Hom\ b\ a] \cdot$
$(\mathrm{s}[Hom\ d\ c,\ Hom\ c\ b] \otimes Hom\ b\ a))$
proof −
have $(\mathrm{s}[Hom\ c\ b,\ Hom\ b\ a] \otimes Hom\ d\ c) \cdot$
$\mathrm{s}[Hom\ d\ c,\ Hom\ c\ b \otimes Hom\ b\ a] \cdot$
$\mathrm{a}[Hom\ d\ c,\ Hom\ c\ b,\ Hom\ b\ a] =$
$\mathrm{a}^{-1}[Hom\ b\ a,\ Hom\ c\ b,\ Hom\ d\ c] \cdot$
$\mathrm{s}[Hom\ c\ b \otimes Hom\ d\ c,\ Hom\ b\ a] \cdot$
$(\mathrm{s}[Hom\ d\ c,\ Hom\ c\ b] \otimes Hom\ b\ a)$
proof −
have *1*: $\mathrm{s}[Hom\ d\ c,\ Hom\ c\ b \otimes Hom\ b\ a] \cdot$
$\mathrm{a}[Hom\ d\ c,\ Hom\ c\ b,\ Hom\ b\ a] =$
$\mathrm{a}^{-1}[Hom\ c\ b,\ Hom\ b\ a,\ Hom\ d\ c] \cdot$

(*Hom c b* ⊗ s[*Hom d c*, *Hom b a*]) ·
a[*Hom c b*, *Hom d c*, *Hom b a*] ·
(s[*Hom d c*, *Hom c b*] ⊗ *Hom b a*)
proof −
have s[*Hom d c*, *Hom c b* ⊗ *Hom b a*] ·
a[*Hom d c*, *Hom c b*, *Hom b a*] =
(a^{-1}[*Hom c b*, *Hom b a*, *Hom d c*] ·
a[*Hom c b*, *Hom b a*, *Hom d c*]) ·
s[*Hom d c*, *Hom c b* ⊗ *Hom b a*] ·
a[*Hom d c*, *Hom c b*, *Hom b a*]
using *a b c d comp-assoc-assoc′(2) comp-cod-arr* **by** *simp*
also have ... =
a^{-1}[*Hom c b*, *Hom b a*, *Hom d c*] ·
a[*Hom c b*, *Hom b a*, *Hom d c*] ·
s[*Hom d c*, *Hom c b* ⊗ *Hom b a*] ·
a[*Hom d c*, *Hom c b*, *Hom b a*]
using *comp-assoc* **by** *simp*
also have ... =
a^{-1}[*Hom c b*, *Hom b a*, *Hom d c*] ·
(*Hom c b* ⊗ s[*Hom d c*, *Hom b a*]) ·
a[*Hom c b*, *Hom d c*, *Hom b a*] ·
(s[*Hom d c*, *Hom c b*] ⊗ *Hom b a*)
using *a b c d assoc-coherence EC.ide-Hom* **by** *auto*
finally show *?thesis* **by** *blast*
qed
have *2*: (s[*Hom c b*, *Hom b a*] ⊗ *Hom d c*) ·
a^{-1}[*Hom c b*, *Hom b a*, *Hom d c*] ·
(*Hom c b* ⊗ s[*Hom d c*, *Hom b a*]) =
a^{-1}[*Hom b a*, *Hom c b*, *Hom d c*] ·
s[*Hom c b* ⊗ *Hom d c*, *Hom b a*] ·
inv a[*Hom c b*, *Hom d c*, *Hom b a*]
proof −
have (s[*Hom c b*, *Hom b a*] ⊗ *Hom d c*) ·
a^{-1}[*Hom c b*, *Hom b a*, *Hom d c*] ·
(*Hom c b* ⊗ s[*Hom d c*, *Hom b a*]) =
inv ((*Hom c b* ⊗ s[*Hom b a*, *Hom d c*]) ·
a[*Hom c b*, *Hom b a*, *Hom d c*] ·
(s[*Hom b a*, *Hom c b*] ⊗ *Hom d c*))
proof −
have *inv* ((*Hom c b* ⊗ s[*Hom b a*, *Hom d c*]) ·
a[*Hom c b*, *Hom b a*, *Hom d c*] ·
(s[*Hom b a*, *Hom c b*] ⊗ *Hom d c*)) =
inv (a[*Hom c b*, *Hom b a*, *Hom d c*] ·
(s[*Hom b a*, *Hom c b*] ⊗ *Hom d c*)) ·
inv (*Hom c b* ⊗ s[*Hom b a*, *Hom d c*])
using *a b c d EC.ide-Hom*
inv-comp [*of* a[*Hom c b*, *Hom b a*, *Hom d c*] ·
(s[*Hom b a*, *Hom c b*] ⊗ *Hom d c*)
Hom c b ⊗ s[*Hom b a*, *Hom d c*]]

by *fastforce*
also have ... =
(*inv* (s[*Hom b a*, *Hom c b*] ⊗ *Hom d c*) ·
a^{-1}[*Hom c b*, *Hom b a*, *Hom d c*]) ·
inv (*Hom c b* ⊗ s[*Hom b a*, *Hom d c*])
using *a b c d EC.ide-Hom inv-comp* **by** *simp*
also have ... =
((s[*Hom c b*, *Hom b a*] ⊗ *Hom d c*) ·
a^{-1}[*Hom c b*, *Hom b a*, *Hom d c*]) ·
(*Hom c b* ⊗ s[*Hom d c*, *Hom b a*])

using *a b c d sym-inverse inverse-unique*
apply *auto*[*1*]
by (*metis* ∗)
finally show *?thesis*
using *comp-assoc* **by** *simp*
qed
also have ... =
inv (a[*Hom c b*, *Hom d c*, *Hom b a*] ·
s[*Hom b a*, *Hom c b* ⊗ *Hom d c*] ·
a[*Hom b a*, *Hom c b*, *Hom d c*])
using *a b c d assoc-coherence EC.ide-Hom* **by** *auto*
also have ... =
a^{-1}[*Hom b a*, *Hom c b*, *Hom d c*] ·
inv s[*Hom b a*, *Hom c b* ⊗ *Hom d c*] ·
a^{-1}[*Hom c b*, *Hom d c*, *Hom b a*]
using *a b c d EC.ide-Hom inv-comp inv-tensor comp-assoc*
isos-compose
by *auto*
also have ... =
a^{-1}[*Hom b a*, *Hom c b*, *Hom d c*] ·
s[*Hom c b* ⊗ *Hom d c*, *Hom b a*] ·
a^{-1}[*Hom c b*, *Hom d c*, *Hom b a*]
using *a b c d sym-inverse inv-is-inverse inverse-unique*
by (*metis tensor-preserves-ide EC.ide-Hom*)
finally show *?thesis* **by** *blast*
qed
hence (s[*Hom c b*, *Hom b a*] ⊗ *Hom d c*) ·
a^{-1}[*Hom c b*, *Hom b a*, *Hom d c*] ·
(*Hom c b* ⊗ s[*Hom d c*, *Hom b a*]) ·
a[*Hom c b*, *Hom d c*, *Hom b a*] =
a^{-1}[*Hom b a*, *Hom c b*, *Hom d c*] ·
s[*Hom c b* ⊗ *Hom d c*, *Hom b a*] ·
inv a[*Hom c b*, *Hom d c*, *Hom b a*] ·
a[*Hom c b*, *Hom d c*, *Hom b a*]
by (*metis comp-assoc*)
hence *3*: (s[*Hom c b*, *Hom b a*] ⊗ *Hom d c*) ·
a^{-1}[*Hom c b*, *Hom b a*, *Hom d c*] ·
(*Hom c b* ⊗ s[*Hom d c*, *Hom b a*]) ·

```
                a[Hom c b, Hom d c, Hom b a] =
             a⁻¹[Hom b a, Hom c b, Hom d c] ·
              s[Hom c b ⊗ Hom d c, Hom b a]
      using a b c comp-arr-dom d by fastforce
    have (s[Hom c b, Hom b a] ⊗ Hom d c) ·
            s[Hom d c, Hom c b ⊗ Hom b a] ·
             a[Hom d c, Hom c b, Hom b a] =
          (s[Hom c b, Hom b a] ⊗ Hom d c) ·
            a⁻¹[Hom c b, Hom b a, Hom d c] ·
              (Hom c b ⊗ s[Hom d c, Hom b a]) ·
                a[Hom c b, Hom d c, Hom b a] ·
                  (s[Hom d c, Hom c b] ⊗ Hom b a)
      using 1 by simp
    also have ... =
          ((s[Hom c b, Hom b a] ⊗ Hom d c) ·
            a⁻¹[Hom c b, Hom b a, Hom d c] ·
              (Hom c b ⊗ s[Hom d c, Hom b a]) ·
                a[Hom c b, Hom d c, Hom b a]) ·
                  (s[Hom d c, Hom c b] ⊗ Hom b a)
      using comp-assoc by simp
    also have ... =
          (a⁻¹[Hom b a, Hom c b, Hom d c] ·
            s[Hom c b ⊗ Hom d c, Hom b a]) ·
              (s[Hom d c, Hom c b] ⊗ Hom b a)
      using 3 by simp
    also have ... =
          a⁻¹[Hom b a, Hom c b, Hom d c] ·
            s[Hom c b ⊗ Hom d c, Hom b a] ·
              (s[Hom d c, Hom c b] ⊗ Hom b a)
      using comp-assoc by simp
    finally show ?thesis by simp
  qed
  thus ?thesis by auto
qed
finally have RHS: Comp_op a c d ·
                    (Hom d c ⊗ Comp_op a b c) ·
                      a[Hom d c, Hom c b, Hom b a] =
                  (Comp d c a · (Comp c b a ⊗ Hom d c)) ·
                    (a⁻¹[Hom b a, Hom c b, Hom d c] ·
                      s[Hom c b ⊗ Hom d c, Hom b a] ·
                        (s[Hom d c, Hom c b] ⊗ Hom b a))
  by blast
show Comp_op a b d · (Comp_op b c d ⊗ Hom b a) =
      Comp_op a c d · (Hom d c ⊗ Comp_op a b c) ·
        a[Hom d c, Hom c b, Hom b a]
  using LHS RHS by simp
qed
qed
```

```
end
```

2.3.1 Relation between $(-^{op})_0$ and $(-_0)^{op}$

Kelly (comment before (1.22)) claims, for a category A enriched in a symmetric monoidal category, that we have $(A^{op})_0 = (A_0)^{op}$. This point becomes somewhat confusing, as it depends on the particular formalization one adopts for the notion of "category".

As we can see from the next two facts (*Op-UC-hom-char* and *UC-Op-hom-char*), the hom-sets *Op.UC.hom a b* and *UC.Op.hom a b* are both obtained by using *UC.MkArr* to "tag" elements of *hom* $\mathcal{I}$ (*Hom* (*UC.Dom b*) (*UC.Dom a*)) with *UC.Dom a* and *UC.Dom b*. These two hom-sets are formally distinct if (as is the case for us), the arrows of a category are regarded as containing information about their domain and codomain, so that the hom-sets are disjoint. On the other hand, if one regards a category as a collection of mappings that assign to each pair of objects *a* and *b* a corresponding set *hom a b*, then the hom-sets *Op.UC.hom a b* and *UC.Op.hom a b* could be arranged to be equal, as Kelly suggests.

```
locale category-enriched-in-symmetric-monoidal-category =
  symmetric-monoidal-category +
  enriched-category
begin

  interpretation elementary-symmetric-monoidal-category
                C tensor unity lunit runit assoc sym
    using induces-elementary-symmetric-monoidal-category_CMC by blast

  interpretation Op: opposite-enriched-category C T α ι σ Obj Hom Id Comp ..
  interpretation Op0: underlying-category C T α ι Obj Op.Hom_op Id Op.Comp_op
    ..

  interpretation UC: underlying-category C T α ι Obj Hom Id Comp ..
  interpretation UC.Op: dual-category UC.comp ..

  lemma Op-UC-hom-char:
  assumes UC.ide a and UC.ide b
  shows Op0.hom a b =
        UC.MkArr (UC.Dom a) (UC.Dom b) `
          hom 𝓘 (Hom (UC.Dom b) (UC.Dom a))
    using assms Op0.hom-char [of UC.Dom a UC.Dom b]
          UC.ide-char [of a] UC.ide-char [of b] UC.arr-char
    by force

  lemma UC-Op-hom-char:
  assumes UC.ide a and UC.ide b
  shows UC.Op.hom a b =
        UC.MkArr (UC.Dom b) (UC.Dom a) `
```

hom $\mathcal{I}$ (*Hom* (*UC.Dom b*) (*UC.Dom a*))
using *assms UC.Op.hom-char UC.hom-char* [*of UC.Dom b UC.Dom a*]
UC.ide-char$_{CC}$
by *simp*

abbreviation *toUCOp*
where *toUCOp f* ≡ *if* Op_0*.arr f*
then UC.MkArr (Op_0*.Cod f*) (Op_0*.Dom f*) (Op_0*.Map f*)
else UC.Op.null

sublocale *toUCOp*: *functor* Op_0*.comp UC.Op.comp toUCOp*
proof
show ⋀*f*. ¬ Op_0*.arr f* ⟹ *toUCOp f* = *UC.Op.null*
by *simp*
show *1*: ⋀*f*. Op_0*.arr f* ⟹ *UC.Op.arr* (*toUCOp f*)
using Op_0*.arr-char* **by** *auto*
show ⋀*f*. Op_0*.arr f* ⟹ *UC.Op.dom* (*toUCOp f*) = *toUCOp* (Op_0*.dom f*)
using *1* **by** *simp*
show ⋀*f*. Op_0*.arr f* ⟹ *UC.Op.cod* (*toUCOp f*) = *toUCOp* (Op_0*.cod f*)
using *1* **by** *simp*
show ⋀*g f*. Op_0*.seq g f* ⟹
toUCOp (Op_0*.comp g f*) = *UC.Op.comp* (*toUCOp g*) (*toUCOp f*)
proof −
fix *f g*
assume *fg*: Op_0*.seq g f*
show *toUCOp* (Op_0*.comp g f*) = *UC.Op.comp* (*toUCOp g*) (*toUCOp f*)
proof (*intro UC.arr-eqI*)
show *UC.arr* (*toUCOp* (Op_0*.comp g f*))
using *1 fg UC.Op.arr-char* **by** *blast*
show *2*: *UC.arr* (*UC.Op.comp* (*toUCOp g*) (*toUCOp f*))
using *1* Op_0*.seq-char UC.seq-char fg* **by** *force*
show Op_0*.Dom* (*toUCOp* (Op_0*.comp g f*)) =
Op_0*.Dom* (*UC.Op.comp* (*toUCOp g*) (*toUCOp f*))
using *1 2 fg* Op_0*.seq-char* **by** *fastforce*
show Op_0*.Cod* (*toUCOp* (Op_0*.comp g f*)) =
Op_0*.Cod* (*UC.Op.comp* (*toUCOp g*) (*toUCOp f*))
using *1 2 fg* Op_0*.seq-char* **by** *fastforce*
show Op_0*.Map* (*toUCOp* (Op_0*.comp g f*)) =
Op_0*.Map* (*UC.Op.comp* (*toUCOp g*) (*toUCOp f*))
proof −
have Op_0*.Map* (*toUCOp* (Op_0*.comp g f*)) =
Op.Comp$_{op}$ (*UC.Dom f*) (*UC.Dom g*) (*UC.Cod g*) ·
(*UC.Map g* ⊗ *UC.Map f*) · ι^{-1}
using *1 2 fg* Op_0*.seq-char* **by** *auto*
also have ... = *Comp* (Op_0*.Cod g*) (Op_0*.Dom g*) (Op_0*.Dom f*) ·
(s[*Hom* (Op_0*.Cod g*) (Op_0*.Dom g*),
Hom (Op_0*.Dom g*) (Op_0*.Dom f*)] ·
(Op_0*.Map g* ⊗ Op_0*.Map f*)) · ι^{-1}
using *comp-assoc* **by** *simp*

also have ... = *Comp* (*Op*$_0$.*Cod g*) (*Op*$_0$.*Dom g*) (*Op*$_0$.*Dom f*) ·
((*Op*$_0$.*Map f* ⊗ *Op*$_0$.*Map g*) · s[$\mathcal{I}$, $\mathcal{I}$]) · ι^{-1}
using *fg Op*$_0$.*seq-char Op*$_0$.*arr-char sym-naturality*
by (*metis* (*no-types*, *lifting*) *in-homE mem-Collect-eq*)
also have ... = *Comp* (*Op*$_0$.*Cod g*) (*Op*$_0$.*Dom g*) (*Op*$_0$.*Dom f*) ·
(*Op*$_0$.*Map f* ⊗ *Op*$_0$.*Map g*) · s[$\mathcal{I}$, $\mathcal{I}$] · ι^{-1}
using *comp-assoc* **by** *simp*
also have ... = *Comp* (*Op*$_0$.*Cod g*) (*Op*$_0$.*Dom g*) (*Op*$_0$.*Dom f*) ·
(*Op*$_0$.*Map f* ⊗ *Op*$_0$.*Map g*) · ι^{-1}
using *sym-inv-unit* ι*-def monoidal-category-axioms*
by (*simp add*: *monoidal-category.unitor-coincidence*(*1*))
finally have *Op*$_0$.*Map* (*toUCOp* (*Op*$_0$.*comp g f*)) =
Comp (*Op*$_0$.*Cod g*) (*Op*$_0$.*Dom g*) (*Op*$_0$.*Dom f*) ·
(*Op*$_0$.*Map f* ⊗ *Op*$_0$.*Map g*) · ι^{-1}
by *blast*
also have ... = *Op*$_0$.*Map* (*UC*.*Op*.*comp* (*toUCOp g*) (*toUCOp f*))
using *fg 2* **by** *auto*
finally show *?thesis* **by** *blast*
qed
qed
qed
qed

lemma *functor-toUCOp*:
shows *functor Op*$_0$.*comp UC*.*Op*.*comp toUCOp*
..

abbreviation *toOp*$_0$
where *toOp*$_0$ *f* ≡ *if UC*.*Op*.*arr f*
then Op$_0$.*MkArr* (*UC*.*Cod f*) (*UC*.*Dom f*) (*UC*.*Map f*)
else Op$_0$.*null*

sublocale *toOp*$_0$: *functor UC*.*Op*.*comp Op*$_0$.*comp toOp*$_0$
proof
show ⋀*f*. ¬ *UC*.*Op*.*arr f* ⟹ *toOp*$_0$ *f* = *Op*$_0$.*null*
by *simp*
show *1*: ⋀*f*. *UC*.*Op*.*arr f* ⟹ *Op*$_0$.*arr* (*toOp*$_0$ *f*)
using *UC*.*arr-char* **by** *simp*
show ⋀*f*. *UC*.*Op*.*arr f* ⟹ *Op*$_0$.*dom* (*toOp*$_0$ *f*) = *toOp*$_0$ (*UC*.*Op*.*dom f*)
using *1* **by** *auto*
show ⋀*f*. *UC*.*Op*.*arr f* ⟹ *Op*$_0$.*cod* (*toOp*$_0$ *f*) = *toOp*$_0$ (*UC*.*Op*.*cod f*)
using *1* **by** *auto*
show ⋀*g f*. *UC*.*Op*.*seq g f* ⟹
toOp$_0$ (*UC*.*Op*.*comp g f*) = *Op*$_0$.*comp* (*toOp*$_0$ *g*) (*toOp*$_0$ *f*)
proof −
fix *f g*
assume *fg*: *UC*.*Op*.*seq g f*
show *toOp*$_0$ (*UC*.*Op*.*comp g f*) = *Op*$_0$.*comp* (*toOp*$_0$ *g*) (*toOp*$_0$ *f*)
proof (*intro Op*$_0$.*arr-eqI*)

show Op_0.*arr* (*toOp*$_0$ (*UC.Op.comp g f*))
  using *fg 1* **by** *blast*
show *2*: Op_0.*seq* (*toOp*$_0$ *g*) (*toOp*$_0$ *f*)
  using *fg 1 UC.seq-char UC.arr-char* Op_0.*seq-char* **by** *fastforce*
show Op_0.*Dom* (*toOp*$_0$ (*UC.Op.comp g f*)) =
    Op_0.*Dom* (Op_0.*comp* (*toOp*$_0$ *g*) (*toOp*$_0$ *f*))
  using *fg 1 2* Op_0.*dom-char* Op_0.*cod-char UC.seq-char* Op_0.*seq-char*
  by *auto*
show Op_0.*Cod* (*toOp*$_0$ (*UC.Op.comp g f*)) =
    Op_0.*Cod* (Op_0.*comp* (*toOp*$_0$ *g*) (*toOp*$_0$ *f*))
  using *fg 1 2* Op_0.*dom-char* Op_0.*cod-char UC.seq-char* Op_0.*seq-char*
  by *auto*
show Op_0.*Map* (*toOp*$_0$ (*UC.Op.comp g f*)) =
    Op_0.*Map* (Op_0.*comp* (*toOp*$_0$ *g*) (*toOp*$_0$ *f*))
proof −
  have Op_0.*Map* (Op_0.*comp* (*toOp*$_0$ *g*) (*toOp*$_0$ *f*)) =
      Op.Comp$_{op}$ (Op_0.*Dom* (*toOp*$_0$ *f*)) (Op_0.*Dom* (*toOp*$_0$ *g*))
       (Op_0.*Cod* (*toOp*$_0$ *g*)) ·
       (Op_0.*Map* (*toOp*$_0$ *g*) ⊗ Op_0.*Map* (*toOp*$_0$ *f*)) · *inv ι*
    using *fg 1 2 UC.seq-char* **by** *auto*
  also have ... =
        Comp (Op_0.*Dom g*) (Op_0.*Cod g*) (Op_0.*Cod f*) ·
         (s[*Hom* (Op_0.*Dom g*) (Op_0.*Cod g*),
           Hom (Op_0.*Cod g*) (Op_0.*Cod f*)] ·
           (Op_0.*Map g* ⊗ Op_0.*Map f*)) · *inv ι*
    using *fg comp-assoc* **by** *auto*
  also have ... =
        Comp (Op_0.*Dom g*) (Op_0.*Cod g*) (Op_0.*Cod f*) ·
         ((Op_0.*Map f* ⊗ Op_0.*Map g*) · s[*unity*, *unity*]) · *inv ι*
    using *fg UC.seq-char UC.arr-char sym-naturality*
    by (*metis* (*no-types*, *lifting*) *in-homE UC.Op.arr-char*
      UC.Op.comp-def mem-Collect-eq)
  also have ... =
        Comp (Op_0.*Dom g*) (Op_0.*Cod g*) (Op_0.*Cod f*) ·
         (Op_0.*Map f* ⊗ Op_0.*Map g*) · s[*unity*, *unity*] · *inv ι*
    using *comp-assoc* **by** *simp*
  also have ... =
        Comp (Op_0.*Dom g*) (Op_0.*Cod g*) (Op_0.*Cod f*) ·
         (Op_0.*Map f* ⊗ Op_0.*Map g*) · *inv ι*
    using *sym-inv-unit ι-def monoidal-category-axioms*
    by (*simp add*: *monoidal-category.unitor-coincidence*(*1*))
  also have ... = Op_0.*Map* (*toOp*$_0$ (*UC.Op.comp g f*))
    using *fg UC.seq-char* **by** *simp*
  finally show *?thesis* **by** *argo*
qed
qed
qed
qed

lemma *functor-toOp*$_0$:
shows *functor UC.Op.comp Op*$_0$*.comp toOp*$_0$
  ..

sublocale *inverse-functors UC.Op.comp Op*$_0$*.comp toUCOp toOp*$_0$
  using *Op*$_0$*.MkArr-Map toUCOp.preserves-reflects-arr Op*$_0$*.is-extensional*
    UC.MkArr-Map toOp$_0$*.preserves-reflects-arr UC.Op.is-extensional*
  by *unfold-locales auto*

lemma *inverse-functors-toUCOp-toOp*$_0$:
shows *inverse-functors UC.Op.comp Op*$_0$*.comp toUCOp toOp*$_0$
  ..

end

2.4 Enriched Hom Functors

Here we exhibit covariant and contravariant hom functors as enriched functors, as in [1] Section 1.6. We don't bother to exhibit them as partial functors of a single two-argument functor, as to do so would require us to define the tensor product of enriched categories; something that would require more technology for proving coherence conditions than we have developed at present.

2.4.1 Covariant Case

locale *covariant-Hom =*
  monoidal-category +

  C: elementary-closed-monoidal-category +
  enriched-category +
fixes *x :: 'o*
assumes *x: x ∈ Obj*
begin

  interpretation *C: enriched-category C T α ι ‹Collect ide› exp C.Id C.Comp*
    using *C.is-enriched-in-itself* **by** *simp*
  interpretation *C: self-enriched-category C T α ι exp eval Curry* **..**

  abbreviation hom_o
  where $hom_o \equiv Hom\ x$

  abbreviation hom_a
  where $hom_a \equiv \lambda b\ c.\ if\ b \in Obj \wedge c \in Obj$
    $then\ Curry[Hom\ b\ c,\ Hom\ x\ b,\ Hom\ x\ c]\ (Comp\ x\ b\ c)$
    $else\ null$

  sublocale *enriched-functor C T α ι*

Obj Hom Id Comp
‹*Collect ide*› *exp C.Id C.Comp*
hom_o hom_a

proof
show $\bigwedge a\ b.\ a \notin Obj \vee b \notin Obj \Longrightarrow hom_a\ a\ b = null$
by *auto*
show $\bigwedge y.\ y \in Obj \Longrightarrow hom_o\ y \in Collect\ ide$
using *x ide-Hom* **by** *auto*
show $*$: $\bigwedge a\ b.\ [\![a \in Obj;\ b \in Obj]\!] \Longrightarrow$
«$hom_a\ a\ b : Hom\ a\ b \rightarrow exp\ (hom_o\ a)\ (hom_o\ b)$»
using *x* **by** *auto*
show $\bigwedge a.\ a \in Obj \Longrightarrow hom_a\ a\ a \cdot Id\ a = C.Id\ (hom_o\ a)$
using *x Comp-Id-Hom Comp-in-hom Id-in-hom C.Id-def C.comp-Curry-arr*
apply *auto*[*1*]
by (*metis ide-Hom*)
show $\bigwedge a\ b\ c.\ [\![a \in Obj;\ b \in Obj;\ c \in Obj]\!] \Longrightarrow$
$C.Comp\ (hom_o\ a)\ (hom_o\ b)\ (hom_o\ c) \cdot$
$(hom_a\ b\ c \otimes hom_a\ a\ b) =$
$hom_a\ a\ c \cdot Comp\ a\ b\ c$
proof −
fix *a b c*
assume *a*: $a \in Obj$ **and** *b*: $b \in Obj$ **and** *c*: $c \in Obj$
have $Uncurry[hom_o\ a,\ hom_o\ c]$
$(C.Comp\ (hom_o\ a)\ (hom_o\ b)\ (hom_o\ c) \cdot (hom_a\ b\ c \otimes hom_a\ a\ b)) =$
$Uncurry[hom_o\ a,\ hom_o\ c]\ (hom_a\ a\ c \cdot Comp\ a\ b\ c)$
proof −
have $Uncurry[hom_o\ a,\ hom_o\ c]$
$(C.Comp\ (hom_o\ a)\ (hom_o\ b)\ (hom_o\ c) \cdot (hom_a\ b\ c \otimes hom_a\ a\ b)) =$
$Uncurry[hom_o\ a,\ hom_o\ c]$
$(Curry[exp\ (hom_o\ b)\ (hom_o\ c) \otimes exp\ (hom_o\ a)\ (hom_o\ b),\ hom_o\ a,$
$hom_o\ c]$
$(eval\ (hom_o\ b)\ (hom_o\ c) \cdot$
$(exp\ (hom_o\ b)\ (hom_o\ c) \otimes eval\ (hom_o\ a)\ (hom_o\ b)) \cdot$
$\mathrm{a}[exp\ (hom_o\ b)\ (hom_o\ c),\ exp\ (hom_o\ a)\ (hom_o\ b),$
$hom_o\ a]) \cdot$
$(hom_a\ b\ c \otimes hom_a\ a\ b))$
using *C.Comp-def* **by** *simp*
also have ... =
$Uncurry[hom_o\ a,\ hom_o\ c]$
$(Curry[Hom\ b\ c \otimes Hom\ a\ b,\ hom_o\ a,\ hom_o\ c]$
$((eval\ (hom_o\ b)\ (hom_o\ c) \cdot$
$(exp\ (hom_o\ b)\ (hom_o\ c) \otimes eval\ (hom_o\ a)\ (hom_o\ b)) \cdot$
$\mathrm{a}[exp\ (hom_o\ b)\ (hom_o\ c),\ exp\ (hom_o\ a)\ (hom_o\ b),$
$hom_o\ a]) \cdot$
$((hom_a\ b\ c \otimes hom_a\ a\ b) \otimes hom_o\ a)))$
proof −
have «$hom_a\ b\ c \otimes hom_a\ a\ b :$
$Hom\ b\ c \otimes Hom\ a\ b \rightarrow$
$exp\ (hom_o\ b)\ (hom_o\ c) \otimes exp\ (hom_o\ a)\ (hom_o\ b)$»

using $x\ a\ b\ c\ *$ **by** *force*
moreover have «*eval* (hom_o *b*) (hom_o *c*) ·
(*exp* (hom_o *b*) (hom_o *c*) ⊗ *eval* (hom_o *a*) (hom_o *b*)) ·
a[*exp* (hom_o *b*) (hom_o *c*), *exp* (hom_o *a*) (hom_o *b*), hom_o *a*]
: (*exp* (hom_o *b*) (hom_o *c*) ⊗ *exp* (hom_o *a*) (hom_o *b*))
⊗ hom_o *a*
→ hom_o *c*»
using *x a b c* **by** *simp*
ultimately show *?thesis*
using *x a b c C.comp-Curry-arr* **by** *simp*
qed
also have ... =
(*eval* (hom_o *b*) (hom_o *c*) ·
(*exp* (hom_o *b*) (hom_o *c*) ⊗ *eval* (hom_o *a*) (hom_o *b*)) ·
a[*exp* (hom_o *b*) (hom_o *c*), *exp* (hom_o *a*) (hom_o *b*), hom_o *a*]) ·
((hom_a *b c* ⊗ hom_a *a b*) ⊗ hom_o *a*)
using *x a b c*
C.Uncurry-Curry
[*of Hom b c* ⊗ *Hom a b* hom_o *a* hom_o *c*
(*eval* (hom_o *b*) (hom_o *c*) ·
(*exp* (hom_o *b*) (hom_o *c*) ⊗ *eval* (hom_o *a*) (hom_o *b*)) ·
a[*exp* (hom_o *b*) (hom_o *c*), *exp* (hom_o *a*) (hom_o *b*), hom_o *a*]) ·
((*Curry*[*Hom b c*, hom_o *b*, hom_o *c*] (*Comp x b c*) ⊗
Curry[*Hom a b*, hom_o *a*, hom_o *b*] (*Comp x a b*))
⊗ hom_o *a*)]
by *fastforce*
also have ... =
eval (hom_o *b*) (hom_o *c*) ·
(*exp* (hom_o *b*) (hom_o *c*) ⊗ *eval* (hom_o *a*) (hom_o *b*)) ·
a[*exp* (hom_o *b*) (hom_o *c*), *exp* (hom_o *a*) (hom_o *b*), hom_o *a*] ·
((hom_a *b c* ⊗ hom_a *a b*) ⊗ hom_o *a*)
by (*simp add*: *comp-assoc*)
also have ... =
eval (hom_o *b*) (hom_o *c*) ·
((*exp* (hom_o *b*) (hom_o *c*) ⊗ *eval* (hom_o *a*) (hom_o *b*)) ·
(hom_a *b c* ⊗ hom_a *a b* ⊗ hom_o *a*)) ·
a[*Hom b c*, *Hom a b*, hom_o *a*]
using *x a b c Comp-in-hom*
assoc-naturality
[*of Curry*[*Hom b c*, hom_o *b*, hom_o *c*] (*Comp x b c*)
Curry[*Hom a b*, hom_o *a*, hom_o *b*] (*Comp x a b*)
hom_o *a*]
using *comp-assoc* **by** *auto*
also have ... =
eval (hom_o *b*) (hom_o *c*) ·
(*exp* (hom_o *b*) (hom_o *c*) ·
hom_a *b c* ⊗ *Uncurry*[hom_o *a*, hom_o *b*] (hom_a *a b*)) ·
a[*Hom b c*, *Hom a b*, hom_o *a*]
using *x a b c Comp-in-hom interchange* **by** *simp*

also have ... =
    eval (*hom*$_o$ *b*) (*hom*$_o$ *c*) ·
      (*exp* (*hom*$_o$ *b*) (*hom*$_o$ *c*) · *hom*$_a$ *b c* ⊗ *Comp x a b*) ·
        a[*Hom b c*, *Hom a b*, *hom*$_o$ *a*]
  using *x a b c C.Uncurry-Curry Comp-in-hom* **by** *auto*
also have ... =
    eval (*hom*$_o$ *b*) (*hom*$_o$ *c*) · (*hom*$_a$ *b c* ⊗ *Comp x a b*) ·
      a[*Hom b c*, *Hom a b*, *hom*$_o$ *a*]
  using *x a b c*
  by (*simp add*: *Comp-in-hom comp-ide-arr*)
also have ... =
    eval (*hom*$_o$ *b*) (*hom*$_o$ *c*) ·
      ((*hom*$_a$ *b c* ⊗ *hom*$_o$ *b*) · (*Hom b c* ⊗ *Comp x a b*)) ·
        a[*Hom b c*, *Hom a b*, *hom*$_o$ *a*]
proof −
  have *seq* (*hom*$_a$ *b c*) (*Hom b c*)
    using *x a b c Comp-in-hom C.Curry-in-hom ide-Hom* **by** *simp*
  moreover have *seq* (*hom*$_o$ *b*) (*Comp x a b*)
    using *x a b c Comp-in-hom* **by** *fastforce*
  ultimately show *?thesis*
    using *x a b c Comp-in-hom C.Curry-in-hom comp-arr-ide*
      comp-ide-arr ide-Hom interchange
    by *metis*
qed
also have ... =
    Uncurry[*hom*$_o$ *b*, *hom*$_o$ *c*] (*hom*$_a$ *b c*) ·
      (*Hom b c* ⊗ *Comp x a b*) ·
        a[*Hom b c*, *Hom a b*, *hom*$_o$ *a*]
  using *comp-assoc* **by** *simp*
also have ... = *Comp x a c* · (*Comp a b c* ⊗ *hom*$_o$ *a*)
  using *x a b c C.Uncurry-Curry Comp-in-hom Comp-assoc* **by** *auto*
also have ... = *Uncurry*[*hom*$_o$ *a*, *hom*$_o$ *c*]
      (*Curry*[*Hom b c* ⊗ *Hom a b*, *hom*$_o$ *a*, *hom*$_o$ *c*]
        (*Comp x a c* · (*Comp a b c* ⊗ *hom*$_o$ *a*)))
  using *x a b c Comp-in-hom comp-assoc*
    C.Uncurry-Curry
      [*of Hom b c* ⊗ *Hom a b hom*$_o$ *a hom*$_o$ *c*
        Comp x a c · (*Comp a b c* ⊗ *hom*$_o$ *a*)]
  by *fastforce*
also have ... = *Uncurry*[*hom*$_o$ *a*, *hom*$_o$ *c*] (*hom*$_a$ *a c* · *Comp a b c*)
  using *x a b c Comp-in-hom*
    C.comp-Curry-arr
      [*of hom*$_o$ *a Comp a b c Hom b c* ⊗ *Hom a b*
        Hom a c Comp x a c hom$_o$ *c*]
  by *auto*
finally show *?thesis* **by** *blast*
qed
hence *Curry*[*Hom b c* ⊗ *Hom a b*, *hom*$_o$ *a*, *hom*$_o$ *c*]
    (*Uncurry*[*hom*$_o$ *a*, *hom*$_o$ *c*]

$(C.Comp\ (hom_o\ a)\ (hom_o\ b)\ (hom_o\ c) \cdot$
$(hom_a\ b\ c \otimes hom_a\ a\ b))) =$
$Curry[Hom\ b\ c \otimes Hom\ a\ b,\ hom_o\ a,\ hom_o\ c]$
$(Uncurry[hom_o\ a,\ hom_o\ c]\ (hom_a\ a\ c \cdot Comp\ a\ b\ c))$
by *simp*
thus $C.Comp\ (hom_o\ a)\ (hom_o\ b)\ (hom_o\ c) \cdot (hom_a\ b\ c \otimes hom_a\ a\ b) =$
$hom_a\ a\ c \cdot Comp\ a\ b\ c$
using *x a b c Comp-in-hom*
C.Curry-Uncurry
$[of\ Hom\ b\ c \otimes Hom\ a\ b\ hom_o\ a\ hom_o\ c\ hom_a\ a\ c \cdot Comp\ a\ b\ c]$
C.Curry-Uncurry
$[of\ Hom\ b\ c \otimes Hom\ a\ b\ hom_o\ a\ hom_o\ c$
$C.Comp\ (hom_o\ a)\ (hom_o\ b)\ (hom_o\ c) \cdot (hom_a\ b\ c \otimes hom_a\ a\ b)]$
by *auto*
qed
qed

lemma *is-enriched-functor*:
shows *enriched-functor C T α ι*
Obj Hom Id Comp
(Collect ide) exp C.Id C.Comp
$hom_o\ hom_a$
..

sublocale C_0: *underlying-category C T α ι ‹Collect ide› exp C.Id C.Comp* **..**
sublocale *UC*: *underlying-category C T α ι Obj Hom Id Comp* **..**
sublocale *UF*: *underlying-functor C T α ι*
Obj Hom Id Comp
‹Collect ide› exp C.Id C.Comp
$hom_o\ hom_a$
..

The following is Kelly's formula (1.31), for the result of applying the ordinary functor underlying the covariant hom functor, to an arrow $g : \mathcal{I} \rightarrow Hom\ b\ c$ of C_0, resulting in an arrow $Hom^{\rightarrow}\ x\ g : Hom\ x\ b \rightarrow Hom\ x\ c$ of C. The point of the result is that this can be expressed explicitly as $Comp\ x\ b\ c \cdot (g \otimes hom_o\ b) \cdot l^{-1}[hom_o\ b]$. This is all very confusing at first, because Kelly identifies C with the underlying category C_0 of C regarded as a self-enriched category, whereas here we cannot ignore the fact that they are merely isomorphic via *C.frmUC*: $UC.comp \rightarrow C_0.comp$. There is also the bother that, for an arrow $g : \mathcal{I} \rightarrow Hom\ b\ c$ of C, the corresponding arrow of the underlying category *UC* has to be formally constructed using *UC.MkArr*, *i.e.* as *UC.MkArr b c g*.

lemma *Kelly-1-31*:
assumes $b \in Obj$ **and** $c \in Obj$ **and** «$g : \mathcal{I} \rightarrow Hom\ b\ c$»
shows $C.frmUC\ (UF.map_0\ (UC.MkArr\ b\ c\ g)) =$
$Comp\ x\ b\ c \cdot (g \otimes hom_o\ b) \cdot l^{-1}[hom_o\ b]$
proof −

have *C.frmUC* (*UF.map*$_0$ (*UC.MkArr b c g*)) =
(*Curry*[*Hom b c*, *hom*$_o$ *b*, *hom*$_o$ *c*] (*Comp x b c*) · *g*) $^\downarrow$[*hom*$_o$ *b*, *hom*$_o$ *c*]
using *assms x ide-Hom UF.map*$_0$*-def*
C.UC.arr-MkArr
[*of Hom x b Hom x c*
Curry[*Hom b c*, *Hom x b*, *Hom x c*] (*Comp x b c*) · *g*]
by *fastforce*
also have ... = *C.Uncurry* (*Hom x b*) (*Hom x c*)
(*Curry*[$\mathcal{I}$, *Hom x b*, *Hom x c*]
(*Comp x b c* · (*g* ⊗ *Hom x b*))) · l^{-1}[*hom*$_o$ *b*]
using *assms x C.comp-Curry-arr C.DN-def*
by (*metis Comp-in-hom C.Curry-simps*(*1*−*2*) *in-homE seqI ide-Hom*)
also have ... = (*Comp x b c* · (*g* ⊗ *Hom x b*)) · l^{-1}[*hom*$_o$ *b*]
using *assms x ide-Hom ide-unity*
C.Uncurry-Curry
[*of* $\mathcal{I}$ *Hom x b Hom x c Comp x b c* · (*g* ⊗ *Hom x b*)]
by *fastforce*
also have ... = *Comp x b c* · (*g* ⊗ *Hom x b*) · l^{-1}[*hom*$_o$ *b*]
using *comp-assoc* **by** *simp*
finally show *?thesis* **by** *blast*
qed

abbreviation *map*$_0$
where *map*$_0$ *b c g* ≡ *Comp x b c* · (*g* ⊗ *Hom x b*) · l^{-1}[*hom*$_o$ *b*]

end

context *elementary-closed-monoidal-category*
begin

lemma *cov-Exp-DN*:
assumes «*g* : $\mathcal{I}$ → *exp a b*»
and *ide a* **and** *ide b* **and** *ide x*
shows *Exp*$^\rightarrow$ *x* (*g* $^\downarrow$[*a*, *b*]) =
(*Curry*[*exp a b*, *exp x a*, *exp x b*] (*Comp x a b*) · *g*) $^\downarrow$[*exp x a*, *exp x b*]
proof −
have (*Curry*[*exp a b*, *exp x a*, *exp x b*] (*Comp x a b*) · *g*) $^\downarrow$[*exp x a*, *exp x b*] =
Uncurry[*exp x a*, *exp x b*]
(*Curry*[$\mathcal{I}$, *exp x a*, *exp x b*] (*Comp x a b* · (*g* ⊗ *exp x a*))) · l^{-1}[*exp x a*]
using *assms DN-def*
comp-Curry-arr
[*of exp x a g* $\mathcal{I}$ *exp a b Comp x a b exp x b*]
by *force*
also have ... = (*Comp x a b* · (*g* ⊗ *exp x a*)) · l^{-1}[*exp x a*]
using *assms Uncurry-Curry* **by** *auto*
also have ... = *Curry*[*exp a b* ⊗ *exp x a*, *x*, *b*]
(*eval a b* · (*exp a b* ⊗ *eval x a*) · a[*exp a b*, *exp x a*, *x*]) ·

```
                    (g ⊗ exp x a) · l⁻¹[exp x a]
      unfolding Comp-def
      using assms comp-assoc by auto
    also have ... = Curry[exp x a, x, b]
                    ((eval a b · (exp a b ⊗ eval x a) · a[exp a b, exp x a, x]) ·
                      ((g ⊗ exp x a) · l⁻¹[exp x a] ⊗ x))
      using assms comp-Curry-arr by auto
    also have ... = Curry[exp x a, x, b]
                    (eval a b · (exp a b ⊗ eval x a) ·
                      (a[exp a b, exp x a, x] · ((g ⊗ exp x a) ⊗ x)) ·
                        (l⁻¹[exp x a] ⊗ x))
     using assms comp-arr-dom comp-cod-arr interchange comp-assoc by fastforce
    also have ... = Curry[exp x a, x, b]
                    (eval a b · (exp a b ⊗ eval x a) ·
                      ((g ⊗ exp x a ⊗ x) · a[ℐ, exp x a, x]) ·
                        (l⁻¹[exp x a] ⊗ x))
      using assms assoc-naturality [of g exp x a x] by auto
    also have ... = Curry[exp x a, x, b]
                    (eval a b · ((exp a b ⊗ eval x a) · (g ⊗ exp x a ⊗ x)) ·
                      a[ℐ, exp x a, x] · (l⁻¹[exp x a] ⊗ x))
      using assms comp-assoc by simp
    also have ... = Curry[exp x a, x, b]
                    (eval a b · ((g ⊗ a) · (ℐ ⊗ eval x a)) ·
                      a[ℐ, exp x a, x] · (l⁻¹[exp x a] ⊗ x))
      using assms comp-arr-dom comp-cod-arr interchange by auto
    also have ... = Curry[exp x a, x, b]
                    (Uncurry[a, b] g · (ℐ ⊗ eval x a) · l⁻¹[exp x a ⊗ x])
      using assms lunit-tensor inv-comp comp-assoc by simp
    also have ... = Exp→ x (g ↓[a, b])
      using assms lunit'-naturality [of eval x a] comp-assoc DN-def by auto
    finally show ?thesis by simp
  qed

end
```

2.4.2 Contravariant Case

```
locale contravariant-Hom =
  symmetric-monoidal-category +
  C: elementary-closed-symmetric-monoidal-category +
  enriched-category +
fixes y :: 'o
assumes y: y ∈ Obj
begin

  interpretation C: enriched-category C T α ι ‹Collect ide› exp C.Id C.Comp
    using C.is-enriched-in-itself by simp
  interpretation C: self-enriched-category C T α ι exp eval Curry ..
```

sublocale *Op*: *opposite-enriched-category C T* α ι σ *Obj Hom Id Comp* **..**

abbreviation hom_o
where $hom_o \equiv \lambda a.\ Hom\ a\ y$

abbreviation hom_a
where $hom_a \equiv \lambda b\ c.\ if\ b \in Obj \wedge c \in Obj$
$then\ Curry[Hom\ c\ b,\ Hom\ b\ y,\ Hom\ c\ y]\ (Op.Comp_{op}\ y\ b\ c)$
$else\ null$

sublocale *enriched-functor C T* α ι
$Obj\ Op.Hom_{op}\ Id\ Op.Comp_{op}$
‹*Collect ide*› *exp C.Id C.Comp*
$hom_o\ hom_a$
proof
show $\bigwedge a\ b.\ a \notin Obj \vee b \notin Obj \Longrightarrow hom_a\ a\ b = null$
by *auto*
show $\bigwedge x.\ x \in Obj \Longrightarrow hom_o\ x \in Collect\ ide$
using *y* **by** *auto*
show $*$: $\bigwedge a\ b.\ [\![a \in Obj;\ b \in Obj]\!] \Longrightarrow$
$«hom_a\ a\ b : Hom\ b\ a \rightarrow exp\ (hom_o\ a)\ (hom_o\ b)»$
using *y C.cnt-Exp-ide C.Curry-in-hom Op.Comp-in-hom* [*of y*] **by** *simp*
show $\bigwedge a.\ a \in Obj \Longrightarrow hom_a\ a\ a \cdot Id\ a = C.Id\ (hom_o\ a)$
using *y Id-in-hom C.Id-def C.comp-Curry-arr Op.Comp-Id-Hom Op.Comp-in-hom*
by *fastforce*
show $\bigwedge a\ b\ c.\ [\![a \in Obj;\ b \in Obj;\ c \in Obj]\!] \Longrightarrow$
$C.Comp\ (hom_o\ a)\ (hom_o\ b)\ (hom_o\ c) \cdot$
$(hom_a\ b\ c \otimes hom_a\ a\ b) =$
$hom_a\ a\ c \cdot Op.Comp_{op}\ a\ b\ c$
proof −
fix *a b c*
assume *a*: $a \in Obj$ **and** *b*: $b \in Obj$ **and** *c*: $c \in Obj$
have $C.Comp\ (hom_o\ a)\ (hom_o\ b)\ (hom_o\ c) \cdot (hom_a\ b\ c \otimes hom_a\ a\ b) =$
$Curry[exp\ (hom_o\ b)\ (hom_o\ c) \otimes exp\ (hom_o\ a)\ (hom_o\ b),$
$hom_o\ a,\ hom_o\ c]$
$(eval\ (hom_o\ b)\ (hom_o\ c) \cdot$
$(exp\ (hom_o\ b)\ (hom_o\ c) \otimes eval\ (hom_o\ a)\ (hom_o\ b)) \cdot$
$\mathrm{a}[exp\ (hom_o\ b)\ (hom_o\ c),\ exp\ (hom_o\ a)\ (hom_o\ b),\ hom_o\ a]) \cdot$
$(hom_a\ b\ c \otimes hom_a\ a\ b)$
using *y a b c comp-assoc C.Comp-def* **by** *simp*
also have ... $= Curry[Hom\ c\ b \otimes Hom\ b\ a,\ hom_o\ a,\ hom_o\ c]$
$((eval\ (hom_o\ b)\ (hom_o\ c) \cdot$
$(exp\ (hom_o\ b)\ (hom_o\ c) \otimes eval\ (hom_o\ a)\ (hom_o\ b)) \cdot$
$\mathrm{a}[exp\ (hom_o\ b)\ (hom_o\ c),\ exp\ (hom_o\ a)\ (hom_o\ b),$
$hom_o\ a]) \cdot$
$((hom_a\ b\ c \otimes hom_a\ a\ b) \otimes hom_o\ a))$
using *y a b c*
C.comp-Curry-arr
[*of Hom a y* $hom_a\ b\ c \otimes hom_a\ a\ b$ *Hom c b* $\otimes$ *Hom b a*

exp (hom_o *b*) (hom_o *c*) ⊗ *exp* (hom_o *a*) (hom_o *b*)
eval (hom_o *b*) (hom_o *c*) ·
(*exp* (hom_o *b*) (hom_o *c*) ⊗ *eval* (hom_o *a*) (hom_o *b*)) ·
a[*exp* (hom_o *b*) (hom_o *c*), *exp* (hom_o *a*) (hom_o *b*), hom_o *a*]
hom_o *c*]
by *fastforce*
also have ... = *Curry*[*Hom c b* ⊗ *Hom b a*, hom_o *a*, hom_o *c*]
(*eval* (hom_o *b*) (hom_o *c*) ·
(*exp* (hom_o *b*) (hom_o *c*) ⊗ *eval* (hom_o *a*) (hom_o *b*)) ·
(hom_a *b c* ⊗ hom_a *a b* ⊗ hom_o *a*) ·
a[*Hom c b*, *Hom b a*, hom_o *a*])
using *y a b c Op.Comp-in-hom comp-assoc*
C.assoc-naturality
[*of Curry*[*Hom c b*, hom_o *b*, hom_o *c*] (*Op.*$Comp_{op}$ *y b c*)
Curry[*Hom b a*, hom_o *a*, hom_o *b*] (*Op.*$Comp_{op}$ *y a b*)
hom_o *a*]
by *auto*
also have ... = *Curry*[*Hom c b* ⊗ *Hom b a*, hom_o *a*, hom_o *c*]
(*eval* (hom_o *b*) (hom_o *c*) ·
(*exp* (hom_o *b*) (hom_o *c*) · hom_a *b c* ⊗
Uncurry[hom_o *a*, hom_o *b*] (hom_a *a b*)) ·
a[*Hom c b*, *Hom b a*, hom_o *a*])
proof −
have *seq* (*exp* (hom_o *b*) (hom_o *c*)) (hom_a *b c*)
using *y a b c* **by** *force*
moreover have *seq* (*eval* (hom_o *a*) (hom_o *b*)) (hom_a *a b* ⊗ hom_o *a*)
using *y a b c* **by** *fastforce*
ultimately show *?thesis*
using *y a b c comp-assoc*
C.interchange
[*of exp* (*Hom b y*) (*Hom c y*) hom_a *b c*
eval (*Hom a y*) (*Hom b y*) hom_a *a b* ⊗ hom_o *a*]
by *metis*
qed
also have ... = *Curry*[*Hom c b* ⊗ *Hom b a*, hom_o *a*, hom_o *c*]
(*eval* (hom_o *b*) (hom_o *c*) ·
(*exp* (hom_o *b*) (hom_o *c*) · hom_a *b c* ⊗ *Op.*$Comp_{op}$ *y a b*) ·
a[*Hom c b*, *Hom b a*, hom_o *a*])
using *y a b c C.Uncurry-Curry Op.Comp-in-hom* **by** *auto*
also have ... = *Curry*[*Hom c b* ⊗ *Hom b a*, hom_o *a*, hom_o *c*]
(*eval* (hom_o *b*) (hom_o *c*) ·
(hom_a *b c* ⊗ *Op.*$Comp_{op}$ *y a b*) ·
a[*Hom c b*, *Hom b a*, hom_o *a*])
using *y a b c*
by (*simp add*: *comp-ide-arr Op.Comp-in-hom*)
also have ... = *Curry*[*Hom c b* ⊗ *Hom b a*, hom_o *a*, hom_o *c*]
(*eval* (hom_o *b*) (hom_o *c*) ·
(hom_a *b c* · *Hom c b* ⊗ hom_o *b* · *Op.*$Comp_{op}$ *y a b*) ·
a[*Hom c b*, *Hom b a*, hom_o *a*])

using $y\ a\ b\ c\ *$
by (*metis Op.Comp-in-hom comp-cod-arr comp-arr-dom in-homE*)
also have ... = $Curry[Hom\ c\ b \otimes Hom\ b\ a,\ hom_o\ a,\ hom_o\ c]$
$(eval\ (hom_o\ b)\ (hom_o\ c) \cdot$
$((hom_a\ b\ c \otimes hom_o\ b) \cdot (Hom\ c\ b \otimes Op.Comp_{op}\ y\ a\ b)) \cdot$
$\mathrm{a}[Hom\ c\ b,\ Hom\ b\ a,\ hom_o\ a])$
using $y\ a\ b\ c\ *$
$C.interchange\ [of\ hom_a\ b\ c\ Hom\ c\ b\ hom_o\ b\ Op.Comp_{op}\ y\ a\ b]$
by (*metis Op.Comp-in-hom ide-Hom ide-char in-homE seqI*)
also have ... = $Curry[Hom\ c\ b \otimes Hom\ b\ a,\ hom_o\ a,\ hom_o\ c]$
$(Uncurry[hom_o\ b,\ hom_o\ c]\ (hom_a\ b\ c) \cdot$
$(Hom\ c\ b \otimes Op.Comp_{op}\ y\ a\ b) \cdot$
$\mathrm{a}[Hom\ c\ b,\ Hom\ b\ a,\ hom_o\ a])$
using *comp-assoc* **by** *simp*
also have ... = $Curry[Hom\ c\ b \otimes Hom\ b\ a,\ hom_o\ a,\ hom_o\ c]$
$(Op.Comp_{op}\ y\ b\ c \cdot$
$(Hom\ c\ b \otimes Op.Comp_{op}\ y\ a\ b) \cdot$
$\mathrm{a}[Hom\ c\ b,\ Hom\ b\ a,\ hom_o\ a])$
using $y\ a\ b\ c\ C.Uncurry\text{-}Curry$
by (*simp add*: *Op.Comp-in-hom*)
also have ... = $Curry[Hom\ c\ b \otimes Hom\ b\ a,\ hom_o\ a,\ hom_o\ c]$
$(Op.Comp_{op}\ y\ a\ c \cdot (Op.Comp_{op}\ a\ b\ c \otimes hom_o\ a))$
using $y\ a\ b\ c\ Op.Comp\text{-}assoc\ [of\ y\ a\ b\ c]$ **by** *simp*
also have ... = $hom_a\ a\ c \cdot Op.Comp_{op}\ a\ b\ c$
using $y\ a\ b\ c\ C.comp\text{-}Curry\text{-}arr\ [of\ Hom\ a\ y\ Op.Comp_{op}\ a\ b\ c]$
ide-Hom Op.Comp-in-hom
by *fastforce*
finally
show $C.Comp\ (hom_o\ a)\ (hom_o\ b)\ (hom_o\ c) \cdot (hom_a\ b\ c \otimes hom_a\ a\ b) =$
$hom_a\ a\ c \cdot Op.Comp_{op}\ a\ b\ c$
by *blast*
qed
qed

lemma *is-enriched-functor*:
shows *enriched-functor* $C\ T\ \alpha\ \iota$
$Obj\ Op.Hom_{op}\ Id\ Op.Comp_{op}$
$(Collect\ ide)\ exp\ C.Id\ C.Comp$
$hom_o\ hom_a$
..

sublocale C_0: *underlying-category* $C\ T\ \alpha\ \iota$ ‹*Collect ide*› *exp* $C.Id\ C.Comp$ **..**
sublocale Op_0: *underlying-category* $C\ T\ \alpha\ \iota\ Obj\ Op.Hom_{op}\ Id\ Op.Comp_{op}$ **..**
sublocale *UF*: *underlying-functor* $C\ T\ \alpha\ \iota$
$Obj\ Op.Hom_{op}\ Id\ Op.Comp_{op}$
‹*Collect ide*› *exp* $C.Id\ C.Comp$
$hom_o\ hom_a$
..

The following is Kelly's formula (1.32) for $Hom^{\leftarrow}\ f\ y : Hom\ b\ y \rightarrow Hom$

a y.

lemma *Kelly-1-32*:
assumes $a \in Obj$ **and** $b \in Obj$ **and** «$f : \mathcal{I} \rightarrow Hom\ a\ b$»
shows $C.frmUC\ (UF.map_0\ (Op_0.MkArr\ b\ a\ f)) =$
$Comp\ a\ b\ y \cdot (Hom\ b\ y \otimes f) \cdot \mathrm{r}^{-1}[hom_o\ b]$
proof −
have $C.frmUC\ (UF.map_0\ (Op_0.MkArr\ b\ a\ f)) =$
$(Curry[Hom\ a\ b,\ hom_o\ b,\ hom_o\ a]\ (Op.Comp_{op}\ y\ b\ a) \cdot f)$
$^{\downarrow}[hom_o\ b,\ hom_o\ a]$
proof −
have $C.UC.arr\ (Op_0.MkArr\ (Hom\ b\ y)\ (Hom\ a\ y)$
$(Curry[Hom\ a\ b,\ Hom\ b\ y,\ Hom\ a\ y]\ (Op.Comp_{op}\ y\ b\ a) \cdot f))$
using *assms y ide-Hom*
apply *auto*[*1*]
using *C.UC.arr-MkArr*
[*of Hom b y Hom a y*
$Curry[Hom\ a\ b,\ hom_o\ b,\ hom_o\ a]\ (Op.Comp_{op}\ y\ b\ a) \cdot f$]
by *blast*
thus *?thesis*
using *assms* $UF.map_0$*-def* Op_0*.arr-MkArr UF.preserves-arr* **by** *auto*
qed
also have *1*: ... $= Curry[\mathcal{I},\ hom_o\ b,\ hom_o\ a]$
$(Op.Comp_{op}\ y\ b\ a \cdot (f \otimes hom_o\ b))\ ^{\downarrow}[hom_o\ b,\ hom_o\ a]$
proof −
have $Curry[Hom\ a\ b,\ Hom\ b\ y,\ Hom\ a\ y]\ (Op.Comp_{op}\ y\ b\ a) \cdot f =$
$Curry[\mathcal{I},\ Hom\ b\ y,\ Hom\ a\ y]\ (Op.Comp_{op}\ y\ b\ a \cdot (f \otimes Hom\ b\ y))$
using *assms y C.comp-Curry-arr* **by** *blast*
thus *?thesis* **by** *simp*
qed
also have ... $= (Op.Comp_{op}\ y\ b\ a \cdot (f \otimes Hom\ b\ y)) \cdot \mathrm{l}^{-1}[hom_o\ b]$
proof −
have $arr\ (Curry[\mathcal{I},\ Hom\ b\ y,\ Hom\ a\ y]$
$(Op.Comp_{op}\ y\ b\ a \cdot (f \otimes Hom\ b\ y)))$
using *assms y ide-Hom C.ide-unity*
by (*metis 1 C.Curry-simps*(*1*−*3*) *C.DN-def C.DN-simps*(*1*) *cod-comp*
dom-comp in-homE not-arr-null seqI Op.Comp-in-hom)
thus *?thesis*
unfolding *C.DN-def*
using *assms y ide-Hom C.ide-unity*
C.Uncurry-Curry
[*of* $\mathcal{I}$ *Hom b y Hom a y* $Op.Comp_{op}\ y\ b\ a \cdot (f \otimes Hom\ b\ y)$]
apply *auto*[*1*]
by *fastforce*
qed
also have ... =
$Comp\ a\ b\ y \cdot (\mathrm{s}[Hom\ a\ b,\ Hom\ b\ y] \cdot (f \otimes Hom\ b\ y)) \cdot \mathrm{l}^{-1}[hom_o\ b]$
using *comp-assoc* **by** *simp*
also have ... $= Comp\ a\ b\ y \cdot ((Hom\ b\ y \otimes f) \cdot \mathrm{s}[\mathcal{I},\ Hom\ b\ y]) \cdot \mathrm{l}^{-1}[hom_o\ b]$
using *assms y C.sym-naturality* [*of f Hom b y*] **by** *auto*

```
    also have ... = Comp a b y · (Hom b y ⊗ f) · s[ℐ, Hom b y] · l⁻¹[hom_o b]
      using comp-assoc by simp
    also have ... = Comp a b y · (Hom b y ⊗ f) · r⁻¹[hom_o b]
    proof -
      have r⁻¹[hom_o b] = inv (l[hom_o b] · s[Hom b y, ℐ])
         using assms y unitor-coherence by simp
      also have ... = s[ℐ, Hom b y] · l⁻¹[hom_o b]
        using assms y
        by (metis C.ide-unity inv-comp-left(1) inverse-unique
            C.iso-lunit C.iso-runit C.sym-inverse C.unitor-coherence
            Op.ide-Hom)
      finally show ?thesis by simp
    qed
    finally show ?thesis by blast
  qed

  abbreviation map₀
  where map₀ a b f ≡ Comp a b y · (Hom b y ⊗ f) · r⁻¹[hom_o b]

end

context elementary-closed-symmetric-monoidal-category
begin

  interpretation enriched-category C T α ι ‹Collect ide› exp Id Comp
    using is-enriched-in-itself by simp
  interpretation self-enriched-category C T α ι exp eval Curry ..

  sublocale Op: opposite-enriched-category C T α ι σ ‹Collect ide› exp Id Comp
    ..

  lemma cnt-Exp-DN:
  assumes «f : ℐ → exp a b»
  and ide a and ide b and ide y
  shows Exp← (f ↓[a, b]) y =
        (Curry[exp a b, exp b y, exp a y] (Op.Comp_op y b a) · f)
          ↓[exp b y, exp a y]
  proof -
    have (Curry[exp a b, exp b y, exp a y] (Op.Comp_op y b a) · f)
           ↓[exp b y, exp a y] =
         Uncurry[exp b y, exp a y]
           (Curry[ℐ, exp b y, exp a y] (Op.Comp_op y b a · (f ⊗ exp b y))) ·
             l⁻¹[exp b y]
      using assms Op.Comp-in-hom DN-def comp-Curry-arr by force
    also have ... = (Op.Comp_op y b a · (f ⊗ exp b y)) · l⁻¹[exp b y]
      using assms Uncurry-Curry by auto
    also have ... = Comp a b y · (s[exp a b, exp b y] · (f ⊗ exp b y)) ·
                    l⁻¹[exp b y]
      using comp-assoc by simp
```

also have ... = *Comp a b y* · ((*exp b y* ⊗ *f*) · s[$\mathcal{I}$, *exp b y*]) · l^{-1}[*exp b y*]
  using *assms sym-naturality* [*of f exp b y*] **by** *auto*
also have ... = *Comp a b y* · (*exp b y* ⊗ *f*) · s[$\mathcal{I}$, *exp b y*] · l^{-1}[*exp b y*]
  using *comp-assoc* **by** *simp*
also have ... = *Comp a b y* · (*exp b y* ⊗ *f*) · r^{-1}[*exp b y*]
proof −
  have r^{-1}[*exp b y*] = *inv* (l[*exp b y*] · s[*exp b y*, $\mathcal{I}$])
    using *assms unitor-coherence* **by** *auto*
  also have ... = *inv* s[*exp b y*, $\mathcal{I}$] · l^{-1}[*exp b y*]
    using *assms inv-comp* **by** *simp*
  also have ... = s[$\mathcal{I}$, *exp b y*] · l^{-1}[*exp b y*]
    using *assms*
    by (*metis ide-exp ide-unity inverse-unique sym-inverse*)
  finally show *?thesis* **by** *simp*
qed
also have ... = *Curry*[*exp b y* ⊗ *exp a b*, *a*, *y*]
    (*eval b y* · (*exp b y* ⊗ *eval a b*) · a[*exp b y*, *exp a b*, *a*]) ·
      (*exp b y* ⊗ *f*) · r^{-1}[*exp b y*]
  unfolding *Comp-def* **by** *simp*
also have ... = *Curry*[*exp b y*, *a*, *y*]
    ((*eval b y* · (*exp b y* ⊗ *eval a b*) · a[*exp b y*, *exp a b*, *a*]) ·
      ((*exp b y* ⊗ *f*) · r^{-1}[*exp b y*] ⊗ *a*))
  using *assms*
      comp-Curry-arr
        [*of a* (*exp b y* ⊗ *f*) · r^{-1}[*exp b y*] *exp b y exp b y* ⊗ *exp a b*
          eval b y · (*exp b y* ⊗ *eval a b*) · a[*exp b y*, *exp a b*, *a*] *y*]
  by *auto*
also have ... = *Curry*[*exp b y*, *a*, *y*]
    ((*eval b y* · (*exp b y* ⊗ *eval a b*) · a[*exp b y*, *exp a b*, *a*]) ·
      (((*exp b y* ⊗ *f*) ⊗ *a*) · (r^{-1}[*exp b y*] ⊗ *a*)))
  using *assms comp-arr-dom comp-cod-arr interchange* **by** *auto*
also have ... = *Curry*[*exp b y*, *a*, *y*]
    (*eval b y* · (*exp b y* ⊗ *eval a b*) ·
      (a[*exp b y*, *exp a b*, *a*] · ((*exp b y* ⊗ *f*) ⊗ *a*)) ·
        (r^{-1}[*exp b y*] ⊗ *a*))
  using *comp-assoc* **by** *simp*
also have ... = *Curry*[*exp b y*, *a*, *y*]
    (*eval b y* · (*exp b y* ⊗ *eval a b*) ·
      ((*exp b y* ⊗ *f* ⊗ *a*) · a[*exp b y*, $\mathcal{I}$, *a*]) ·
        (r^{-1}[*exp b y*] ⊗ *a*))
  using *assms assoc-naturality* [*of exp b y f a*] **by** *auto*
also have ... = *Curry*[*exp b y*, *a*, *y*]
    (*eval b y* ·
      ((*exp b y* ⊗ *eval a b*) · (*exp b y* ⊗ *f* ⊗ *a*)) ·
        a[*exp b y*, $\mathcal{I}$, *a*] · (r^{-1}[*exp b y*] ⊗ *a*))
  using *comp-assoc* **by** *simp*
also have ... = *Curry*[*exp b y*, *a*, *y*]
    (*eval b y* · (*exp b y* ⊗ *Uncurry*[*a*, *b*] *f*) ·
      a[*exp b y*, $\mathcal{I}$, *a*] · (r^{-1}[*exp b y*] ⊗ *a*))

using *assms comp-arr-dom comp-cod-arr interchange* **by** *simp*
also have ... = $Curry[exp\ b\ y,\ a,\ y]$
$(eval\ b\ y \cdot (exp\ b\ y \otimes Uncurry[a,\ b]\ f) \cdot (exp\ b\ y \otimes \mathrm{l}^{-1}[a]))$
proof −
have $exp\ b\ y \otimes \mathrm{l}^{-1}[a] = inv\ ((\mathrm{r}[exp\ b\ y] \otimes a) \cdot \mathrm{a}^{-1}[exp\ b\ y,\ \mathcal{I},\ a])$
using *assms triangle′ inv-inv iso-inv-iso*
by (*metis ide-exp ide-is-iso inv-ide inv-tensor iso-lunit*)
also have ... = $\mathrm{a}[exp\ b\ y,\ \mathcal{I},\ a] \cdot (\mathrm{r}^{-1}[exp\ b\ y] \otimes a)$
using *assms inv-comp* **by** *simp*
finally show *?thesis* **by** *simp*
qed
also have ... = $Curry[exp\ b\ y,\ a,\ y]$
$(eval\ b\ y \cdot (exp\ b\ y \otimes Uncurry[a,\ b]\ f \cdot \mathrm{l}^{-1}[a]))$
using *assms comp-arr-dom comp-cod-arr interchange* **by** *fastforce*
also have ... = $Exp^{\leftarrow}\ (f\ ^{\downarrow}[a,\ b])\ y$
using *assms DN-def* **by** *auto*
finally show *?thesis* **by** *simp*
qed

end

2.5 Enriched Yoneda Lemma

In this section we prove the (weak) Yoneda lemma for enriched categories, as in Kelly, Section 1.9. The weakness is due to the fact that the lemma asserts only a bijection between sets, rather than an isomorphism of objects of the underlying base category.

2.5.1 Preliminaries

The following gives conditions under which τ defined as $\tau\ x = (\mathcal{T}\ x)^{\uparrow}$ yields an enriched natural transformation between enriched functors F and G to the self-enriched base category.

context *elementary-closed-monoidal-category*
begin

lemma *transformation-lam-UP*:
assumes *enriched-functor C T α ι*
$Obj_A\ Hom_A\ Id_A\ Comp_A\ (Collect\ ide)\ exp\ Id\ Comp\ F_o\ F_a$
assumes *enriched-functor C T α ι*
$Obj_A\ Hom_A\ Id_A\ Comp_A\ (Collect\ ide)\ exp\ Id\ Comp\ G_o\ G_a$
and $\bigwedge x.\ x \notin Obj_A \Longrightarrow \mathcal{T}\ x = null$
and $\bigwedge x.\ x \in Obj_A \Longrightarrow «\mathcal{T}\ x : F_o\ x \rightarrow G_o\ x»$
and $\bigwedge a\ b.\ [\![a \in Obj_A;\ b \in Obj_A]\!] \Longrightarrow$
$\mathcal{T}\ b \cdot Uncurry[F_o\ a,\ F_o\ b]\ (F_a\ a\ b) =$
$eval\ (G_o\ a)\ (G_o\ b) \cdot (G_a\ a\ b \otimes \mathcal{T}\ a)$
shows *enriched-natural-transformation C T α ι*

Obj_A Hom_A Id_A $Comp_A$ (*Collect ide*) *exp Id Comp*
F_o F_a G_o G_a ($\lambda x.$ ($\mathcal{T}$ x)$^\uparrow$)
proof −
interpret *F*: *enriched-functor C T* α ι
Obj_A Hom_A Id_A $Comp_A$ ‹*Collect ide*› *exp Id Comp* F_o F_a
using *assms*(*1*) **by** *blast*
interpret *G*: *enriched-functor C T* α ι
Obj_A Hom_A Id_A $Comp_A$ ‹*Collect ide*› *exp Id Comp* G_o G_a
using *assms*(*2*) **by** *blast*
show *?thesis*
proof
show $\bigwedge x.\ x \notin Obj_A \Longrightarrow (\mathcal{T}\ x)^\uparrow = null$
unfolding *UP-def*
using *assms*(*3*) **by** *auto*
show $\bigwedge x.\ x \in Obj_A \Longrightarrow$ «$(\mathcal{T}\ x)^\uparrow : \mathcal{I} \rightarrow exp\ (F_o\ x)\ (G_o\ x)$»
unfolding *UP-def*
using *assms*(*4*)
apply *auto*[*1*]
by *force*
fix *a b*
assume *a*: $a \in Obj_A$ **and** *b*: $b \in Obj_A$
have *1*: «$((\mathcal{T}\ b)^\uparrow \otimes F_a\ a\ b) \cdot \mathrm{l}^{-1}[Hom_A\ a\ b]$
$: Hom_A\ a\ b \rightarrow exp\ (F_o\ b)\ (G_o\ b) \otimes exp\ (F_o\ a)\ (F_o\ b)$»
using *assms*(*4*) [*of b*] *a b UP-DN F.preserves-Hom*
apply (*intro comp-in-homI tensor-in-homI*)
apply *auto*[*5*]
by *fastforce*
have *2*: «$(G_a\ a\ b \otimes (\mathcal{T}\ a)^\uparrow) \cdot \mathrm{r}^{-1}[Hom_A\ a\ b]$
$: Hom_A\ a\ b \rightarrow exp\ (G_o\ a)\ (G_o\ b) \otimes exp\ (F_o\ a)\ (G_o\ a)$»
using *assms*(*4*) [*of a*] *a b UP-DN F.preserves-Obj G.preserves-Hom*
apply (*intro comp-in-homI tensor-in-homI*)
apply *auto*[*5*]
by *fastforce*
have *3*: «$Comp\ (F_o\ a)\ (F_o\ b)\ (G_o\ b) \cdot ((\mathcal{T}\ b)^\uparrow \otimes F_a\ a\ b) \cdot \mathrm{l}^{-1}[Hom_A\ a\ b]$
$: Hom_A\ a\ b \rightarrow exp\ (F_o\ a)\ (G_o\ b)$»
using *a b 1 F.preserves-Obj G.preserves-Obj* **by** *blast*
have *4*: «$Comp\ (F_o\ a)\ (G_o\ a)\ (G_o\ b) \cdot (G_a\ a\ b \otimes (\mathcal{T}\ a)^\uparrow) \cdot \mathrm{r}^{-1}[Hom_A\ a\ b]$
$: Hom_A\ a\ b \rightarrow exp\ (F_o\ a)\ (G_o\ b)$»
using *a b 2 F.preserves-Obj G.preserves-Obj* **by** *blast*

have $Uncurry[F_o\ a,\ G_o\ b]$
$(Comp\ (F_o\ a)\ (F_o\ b)\ (G_o\ b) \cdot$
$((\mathcal{T}\ b)^\uparrow \otimes F_a\ a\ b) \cdot \mathrm{l}^{-1}[Hom_A\ a\ b]) =$
$Uncurry[F_o\ a,\ G_o\ b]$
$(Curry[exp\ (F_o\ b)\ (G_o\ b) \otimes exp\ (F_o\ a)\ (F_o\ b),\ F_o\ a,\ G_o\ b]$
$(eval\ (F_o\ b)\ (G_o\ b) \cdot$
$(exp\ (F_o\ b)\ (G_o\ b) \otimes eval\ (F_o\ a)\ (F_o\ b)) \cdot$
$\mathrm{a}[exp\ (F_o\ b)\ (G_o\ b),\ exp\ (F_o\ a)\ (F_o\ b),\ F_o\ a]) \cdot$
$((\mathcal{T}\ b)^\uparrow \otimes F_a\ a\ b) \cdot \mathrm{l}^{-1}[Hom_A\ a\ b])$

using *a b Comp-def comp-assoc* **by** *auto*
also have ... =
$Uncurry[F_o\ a,\ G_o\ b]$
$(Curry[Hom_A\ a\ b,\ F_o\ a,\ G_o\ b]$
$((eval\ (F_o\ b)\ (G_o\ b)\ \cdot$
$(exp\ (F_o\ b)\ (G_o\ b) \otimes eval\ (F_o\ a)\ (F_o\ b))\ \cdot$
$\mathrm{a}[exp\ (F_o\ b)\ (G_o\ b),\ exp\ (F_o\ a)\ (F_o\ b),\ F_o\ a])\ \cdot$
$(((\mathcal{T}\ b)^{\uparrow} \otimes F_a\ a\ b) \cdot \mathrm{l}^{-1}[Hom_A\ a\ b] \otimes F_o\ a)))$
using *a b 1 F.preserves-Obj G.preserves-Obj comp-Curry-arr* **by** *auto*
also have ... =
$(eval\ (F_o\ b)\ (G_o\ b)\ \cdot$
$(exp\ (F_o\ b)\ (G_o\ b) \otimes eval\ (F_o\ a)\ (F_o\ b))\ \cdot$
$\mathrm{a}[exp\ (F_o\ b)\ (G_o\ b),\ exp\ (F_o\ a)\ (F_o\ b),\ F_o\ a])\ \cdot$
$(((\mathcal{T}\ b)^{\uparrow} \otimes F_a\ a\ b) \cdot \mathrm{l}^{-1}[Hom_A\ a\ b] \otimes F_o\ a)$
using *a b 1 F.preserves-Obj G.preserves-Obj Uncurry-Curry* **by** *auto*
also have ... =
$(eval\ (F_o\ b)\ (G_o\ b)\ \cdot$
$(exp\ (F_o\ b)\ (G_o\ b) \otimes eval\ (F_o\ a)\ (F_o\ b))\ \cdot$
$\mathrm{a}[exp\ (F_o\ b)\ (G_o\ b),\ exp\ (F_o\ a)\ (F_o\ b),\ F_o\ a])\ \cdot$
$((((\mathcal{T}\ b)^{\uparrow} \otimes F_a\ a\ b) \otimes F_o\ a) \cdot (\mathrm{l}^{-1}[Hom_A\ a\ b] \otimes F_o\ a))$
proof −
have *seq* $((\mathcal{T}\ b)^{\uparrow} \otimes F_a\ a\ b)\ \mathrm{l}^{-1}[Hom_A\ a\ b]$
using *assms(4) a b 1 F.preserves-Hom [of a b] UP-DN*
apply (*intro seqI*)
apply *auto[2]*
by (*metis F.A.ide-Hom arrI cod-inv dom-lunit iso-lunit seqE*)
thus *?thesis*
using *assms(3) a b F.preserves-Obj F.preserves-Hom interchange*
by *simp*
qed
also have ... =
$eval\ (F_o\ b)\ (G_o\ b)\ \cdot$
$(exp\ (F_o\ b)\ (G_o\ b) \otimes eval\ (F_o\ a)\ (F_o\ b))\ \cdot$
$(\mathrm{a}[exp\ (F_o\ b)\ (G_o\ b),\ exp\ (F_o\ a)\ (F_o\ b),\ F_o\ a]\ \cdot$
$(((\mathcal{T}\ b)^{\uparrow} \otimes F_a\ a\ b) \otimes F_o\ a)) \cdot (\mathrm{l}^{-1}[Hom_A\ a\ b] \otimes F_o\ a)$
using *comp-assoc* **by** *simp*
also have ... =
$eval\ (F_o\ b)\ (G_o\ b)\ \cdot$
$(exp\ (F_o\ b)\ (G_o\ b) \otimes eval\ (F_o\ a)\ (F_o\ b))\ \cdot$
$(((\mathcal{T}\ b)^{\uparrow} \otimes F_a\ a\ b \otimes F_o\ a)\ \cdot$
$\mathrm{a}[\mathcal{I},\ Hom_A\ a\ b,\ F_o\ a])\ \cdot$
$(\mathrm{l}^{-1}[Hom_A\ a\ b] \otimes F_o\ a)$
using *assms(4) a b F.preserves-Obj F.preserves-Hom*
assoc-naturality [*of* $(\mathcal{T}\ b)^{\uparrow}\ F_a\ a\ b\ F_o\ a$]
by *force*
also have ... =
$eval\ (F_o\ b)\ (G_o\ b)\ \cdot$
$((exp\ (F_o\ b)\ (G_o\ b) \otimes eval\ (F_o\ a)\ (F_o\ b))\ \cdot$
$((\mathcal{T}\ b)^{\uparrow} \otimes F_a\ a\ b \otimes F_o\ a))\ \cdot$

a[$\mathcal{I}$, Hom_A a b, F_o a] · (l^{-1}[Hom_A a b] ⊗ F_o a)
  using *comp-assoc* **by** *simp*
also have ... =
  eval (F_o b) (G_o b) ·
    (($\mathcal{T}$ b)$^\uparrow$ ⊗ *Uncurry*[F_o a, F_o b] (F_a a b)) ·
      a[$\mathcal{I}$, Hom_A a b, F_o a] · (l^{-1}[Hom_A a b] ⊗ F_o a)
proof −
  have *seq* (*exp* (F_o b) (G_o b)) (*UP* ($\mathcal{T}$ b))
    using *assms(4)* b *F.preserves-Obj G.preserves-Obj* **by** *fastforce*
  moreover have *seq* (*eval* (F_o a) (F_o b)) (F_a a b ⊗ F_o a)
    using a b *F.preserves-Obj F.preserves-Hom* **by** *force*
  ultimately show *?thesis*
    using *assms(4)* [*of b*] a b *UP-DN(1) comp-cod-arr interchange* **by** *auto*
qed
also have ... =
  eval (F_o b) (G_o b) ·
    ((($\mathcal{T}$ b)$^\uparrow$ ⊗ F_o b) · ($\mathcal{I}$ ⊗ *Uncurry*[F_o a, F_o b] (F_a a b))) ·
      a[$\mathcal{I}$, Hom_A a b, F_o a] · (l^{-1}[Hom_A a b] ⊗ F_o a)
  using *assms(4)* [*of b*] a b *F.preserves-Obj F.preserves-Hom* [*of a b*]
    comp-arr-dom comp-cod-arr [*of Uncurry*[F_o a, F_o b] (F_a a b)]
    interchange [*of* ($\mathcal{T}$ b)$^\uparrow$ $\mathcal{I}$ F_o b *Uncurry*[F_o a, F_o b] (F_a a b)]
  by *auto*
also have ... =
  Uncurry[F_o b, G_o b] (($\mathcal{T}$ b)$^\uparrow$) ·
    ($\mathcal{I}$ ⊗ *Uncurry*[F_o a, F_o b] (F_a a b)) ·
      a[$\mathcal{I}$, Hom_A a b, F_o a] · (l^{-1}[Hom_A a b] ⊗ F_o a)
  using *comp-assoc* **by** *simp*
also have ... = ($\mathcal{T}$ b · l[F_o b]) ·
    ($\mathcal{I}$ ⊗ *Uncurry*[F_o a, F_o b] (F_a a b)) ·
      a[$\mathcal{I}$, Hom_A a b, F_o a] · (l^{-1}[Hom_A a b] ⊗ F_o a)
proof −
  have *Uncurry*[F_o b, G_o b] (($\mathcal{T}$ b)$^\uparrow$) = $\mathcal{T}$ b · l[F_o b]
    unfolding *UP-def*
    using *assms(4)* a b *Uncurry-Curry*
    apply *simp*
    by (*metis F.preserves-Obj arr-lunit cod-lunit comp-in-homI′ dom-lunit ide-cod ide-unity in-homE mem-Collect-eq*)
  thus *?thesis* **by** *simp*
qed
also have ... = $\mathcal{T}$ b · (l[F_o b] · ($\mathcal{I}$ ⊗ *Uncurry*[F_o a, F_o b] (F_a a b))) ·
    a[$\mathcal{I}$, Hom_A a b, F_o a] · (l^{-1}[Hom_A a b] ⊗ F_o a)
  using *comp-assoc* **by** *simp*
also have ... = $\mathcal{T}$ b · (*Uncurry*[F_o a, F_o b] (F_a a b) · l[Hom_A a b ⊗ F_o a])
·
    a[$\mathcal{I}$, Hom_A a b, F_o a] · (l^{-1}[Hom_A a b] ⊗ F_o a)
  using a b *lunit-naturality* [*of Uncurry*[F_o a, F_o b] (F_a a b)]
    F.preserves-Obj F.preserves-Hom [*of a b*]
  by *auto*
also have ... = $\mathcal{T}$ b · *Uncurry*[F_o a, F_o b] (F_a a b) ·

l[*Hom*$_A$ *a b* ⊗ *F*$_o$ *a*] · a[$\mathcal{I}$, *Hom*$_A$ *a b*, *F*$_o$ *a*] ·
(l^{-1}[*Hom*$_A$ *a b*] ⊗ *F*$_o$ *a*)
using *comp-assoc* **by** *simp*
also have ... = $\mathcal{T}$ *b* · *Uncurry*[*F*$_o$ *a*, *F*$_o$ *b*] (*F*$_a$ *a b*)
proof −
have l[*Hom*$_A$ *a b* ⊗ *F*$_o$ *a*] · a[$\mathcal{I}$, *Hom*$_A$ *a b*, *F*$_o$ *a*] ·
(l^{-1}[*Hom*$_A$ *a b*] ⊗ *F*$_o$ *a*) =
Hom$_A$ *a b* ⊗ *F*$_o$ *a*
proof −
have l[*Hom*$_A$ *a b* ⊗ *F*$_o$ *a*] · a[$\mathcal{I}$, *Hom*$_A$ *a b*, *F*$_o$ *a*] ·
(l^{-1}[*Hom*$_A$ *a b*] ⊗ *F*$_o$ *a*) =
(l[*Hom*$_A$ *a b*] ⊗ *F*$_o$ *a*) · (l^{-1}[*Hom*$_A$ *a b*] ⊗ *F*$_o$ *a*)
using *a b lunit-tensor′* [*of Hom*$_A$ *a b F*$_o$ *a*]
by (*metis F.A.ide-Hom F.preserves-Obj comp-assoc mem-Collect-eq*)
also have ... = l[*Hom*$_A$ *a b*] · l^{-1}[*Hom*$_A$ *a b*] ⊗ *F*$_o$ *a* · *F*$_o$ *a*
using *a b interchange F.preserves-Obj* **by** *force*
also have ... = *Hom*$_A$ *a b* ⊗ *F*$_o$ *a*
using *a b F.preserves-Obj* **by** *auto*
finally show *?thesis* **by** *blast*
qed
thus *?thesis*
using *a b F.preserves-Obj F.preserves-Hom* [*of a b*] *comp-arr-dom*
by *auto*
qed
finally have *LHS*: *Uncurry*[*F*$_o$ *a*, *G*$_o$ *b*]
(*Comp* (*F*$_o$ *a*) (*F*$_o$ *b*) (*G*$_o$ *b*) · (($\mathcal{T}$ *b*)$^{\uparrow}$ ⊗ *F*$_a$ *a b*) ·
l^{-1}[*Hom*$_A$ *a b*]) =
$\mathcal{T}$ *b* · *Uncurry*[*F*$_o$ *a*, *F*$_o$ *b*] (*F*$_a$ *a b*)
by *blast*

have *Uncurry*[*F*$_o$ *a*, *G*$_o$ *b*] (*Comp* (*F*$_o$ *a*) (*G*$_o$ *a*) (*G*$_o$ *b*) ·
(*G*$_a$ *a b* ⊗ ($\mathcal{T}$ *a*)$^{\uparrow}$) · r^{-1}[*Hom*$_A$ *a b*]) =
Uncurry[*F*$_o$ *a*, *G*$_o$ *b*]
(*Curry*[*exp* (*G*$_o$ *a*) (*G*$_o$ *b*) ⊗ *exp* (*F*$_o$ *a*) (*G*$_o$ *a*), *F*$_o$ *a*, *G*$_o$ *b*]
(*eval* (*G*$_o$ *a*) (*G*$_o$ *b*) ·
(*exp* (*G*$_o$ *a*) (*G*$_o$ *b*) ⊗ *eval* (*F*$_o$ *a*) (*G*$_o$ *a*)) ·
a[*exp* (*G*$_o$ *a*) (*G*$_o$ *b*), *exp* (*F*$_o$ *a*) (*G*$_o$ *a*), *F*$_o$ *a*]) ·
(*G*$_a$ *a b* ⊗ ($\mathcal{T}$ *a*)$^{\uparrow}$) · r^{-1}[*Hom*$_A$ *a b*])
using *a b Comp-def comp-assoc* **by** *auto*
also have ... =
Uncurry[*F*$_o$ *a*, *G*$_o$ *b*]
(*Curry*[*Hom*$_A$ *a b*, *F*$_o$ *a*, *G*$_o$ *b*]
((*eval* (*G*$_o$ *a*) (*G*$_o$ *b*) ·
(*exp* (*G*$_o$ *a*) (*G*$_o$ *b*) ⊗ *eval* (*F*$_o$ *a*) (*G*$_o$ *a*)) ·
a[*exp* (*G*$_o$ *a*) (*G*$_o$ *b*), *exp* (*F*$_o$ *a*) (*G*$_o$ *a*), *F*$_o$ *a*]) ·
((*G*$_a$ *a b* ⊗ ($\mathcal{T}$ *a*)$^{\uparrow}$) · r^{-1}[*Hom*$_A$ *a b*] ⊗ *F*$_o$ *a*)))
using *assms*(*3*) *a b 2 F.preserves-Obj G.preserves-Obj comp-Curry-arr*
by *auto*
also have ... =

$(\mathit{eval}\ (G_o\ a)\ (G_o\ b) \cdot$
$(\mathit{exp}\ (G_o\ a)\ (G_o\ b) \otimes \mathit{eval}\ (F_o\ a)\ (G_o\ a)) \cdot$
$\mathrm{a}[\mathit{exp}\ (G_o\ a)\ (G_o\ b),\ \mathit{exp}\ (F_o\ a)\ (G_o\ a),\ F_o\ a]) \cdot$
$((G_a\ a\ b \otimes (\mathcal{T}\ a)^{\uparrow}) \cdot \mathrm{r}^{-1}[\mathit{Hom}_A\ a\ b] \otimes F_o\ a)$
using *assms(3) a b 2 F.preserves-Obj G.preserves-Obj Uncurry-Curry*
by *auto*
also have ... =
$(\mathit{eval}\ (G_o\ a)\ (G_o\ b) \cdot$
$(\mathit{exp}\ (G_o\ a)\ (G_o\ b) \otimes \mathit{eval}\ (F_o\ a)\ (G_o\ a)) \cdot$
$\mathrm{a}[\mathit{exp}\ (G_o\ a)\ (G_o\ b),\ \mathit{exp}\ (F_o\ a)\ (G_o\ a),\ F_o\ a]) \cdot$
$(((G_a\ a\ b \otimes (\mathcal{T}\ a)^{\uparrow}) \otimes F_o\ a) \cdot (\mathrm{r}^{-1}[\mathit{Hom}_A\ a\ b] \otimes F_o\ a))$
using *assms(4) a b F.preserves-Obj G.preserves-Hom*
interchange $[\mathit{of}\ G_a\ a\ b \otimes (\mathcal{T}\ a)^{\uparrow}\ \mathrm{r}^{-1}[\mathit{Hom}_A\ a\ b]\ F_o\ a\ F_o\ a]$
by *fastforce*
also have ... =
$\mathit{eval}\ (G_o\ a)\ (G_o\ b) \cdot$
$(\mathit{exp}\ (G_o\ a)\ (G_o\ b) \otimes \mathit{eval}\ (F_o\ a)\ (G_o\ a)) \cdot$
$(\mathrm{a}[\mathit{exp}\ (G_o\ a)\ (G_o\ b),\ \mathit{exp}\ (F_o\ a)\ (G_o\ a),\ F_o\ a] \cdot$
$((G_a\ a\ b \otimes (\mathcal{T}\ a)^{\uparrow}) \otimes F_o\ a)) \cdot (\mathrm{r}^{-1}[\mathit{Hom}_A\ a\ b] \otimes F_o\ a)$
using *comp-assoc* **by** *simp*
also have ... =
$\mathit{eval}\ (G_o\ a)\ (G_o\ b) \cdot$
$(\mathit{exp}\ (G_o\ a)\ (G_o\ b) \otimes \mathit{eval}\ (F_o\ a)\ (G_o\ a)) \cdot$
$((G_a\ a\ b \otimes (\mathcal{T}\ a)^{\uparrow} \otimes F_o\ a) \cdot$
$\mathrm{a}[\mathit{Hom}_A\ a\ b,\ \mathcal{I},\ F_o\ a]) \cdot$
$(\mathrm{r}^{-1}[\mathit{Hom}_A\ a\ b] \otimes F_o\ a)$
using *assms(4) a b F.preserves-Obj G.preserves-Hom*
assoc-naturality $[\mathit{of}\ G_a\ a\ b\ (\mathcal{T}\ a)^{\uparrow}\ F_o\ a]$
by *fastforce*
also have ... =
$\mathit{eval}\ (G_o\ a)\ (G_o\ b) \cdot$
$((\mathit{exp}\ (G_o\ a)\ (G_o\ b) \otimes \mathit{eval}\ (F_o\ a)\ (G_o\ a)) \cdot$
$(G_a\ a\ b \otimes (\mathcal{T}\ a)^{\uparrow} \otimes F_o\ a)) \cdot$
$\mathrm{a}[\mathit{Hom}_A\ a\ b,\ \mathcal{I},\ F_o\ a] \cdot (\mathrm{r}^{-1}[\mathit{Hom}_A\ a\ b] \otimes F_o\ a)$
using *comp-assoc* **by** *simp*
also have ... =
$\mathit{eval}\ (G_o\ a)\ (G_o\ b) \cdot$
$(G_a\ a\ b \otimes \mathit{Uncurry}[F_o\ a,\ G_o\ a]\ ((\mathcal{T}\ a)^{\uparrow})) \cdot$
$\mathrm{a}[\mathit{Hom}_A\ a\ b,\ \mathcal{I},\ F_o\ a] \cdot (\mathrm{r}^{-1}[\mathit{Hom}_A\ a\ b] \otimes F_o\ a)$
using *assms(4) a b F.preserves-Obj G.preserves-Obj*
G.preserves-Hom [of a b] comp-cod-arr interchange
by *fastforce*
also have ... =
$\mathit{eval}\ (G_o\ a)\ (G_o\ b) \cdot$
$((G_a\ a\ b \otimes G_o\ a) \cdot (\mathit{Hom}_A\ a\ b \otimes \mathit{Uncurry}[F_o\ a,\ G_o\ a]\ ((\mathcal{T}\ a)^{\uparrow}))) \cdot$
$\mathrm{a}[\mathit{Hom}_A\ a\ b,\ \mathcal{I},\ F_o\ a] \cdot (\mathrm{r}^{-1}[\mathit{Hom}_A\ a\ b] \otimes F_o\ a)$
proof −
have *seq* $(G_o\ a)\ (\mathit{Uncurry}[F_o\ a,\ G_o\ a]\ ((\mathcal{T}\ a)^{\uparrow}))$
using *assms(4) [of a] a b F.preserves-Obj G.preserves-Obj* **by** *auto*

moreover have $G_o\ a \cdot Uncurry[F_o\ a,\ G_o\ a]\ ((\mathcal{T}\ a)^{\uparrow}) =$
$Uncurry[F_o\ a,\ G_o\ a]\ ((\mathcal{T}\ a)^{\uparrow})$
using *a b F.preserves-Obj G.preserves-Obj calculation(1) comp-ide-arr*
by *blast*
ultimately show *?thesis*
using *assms(3) a b G.preserves-Hom [of a b] interchange comp-arr-dom*
by *auto*
qed
also have ... =
$Uncurry[G_o\ a,\ G_o\ b]\ (G_a\ a\ b) \cdot$
$(Hom_A\ a\ b \otimes Uncurry[F_o\ a,\ G_o\ a]\ ((\mathcal{T}\ a)^{\uparrow})) \cdot$
$\mathrm{a}[Hom_A\ a\ b,\ \mathcal{I},\ F_o\ a] \cdot (\mathrm{r}^{-1}[Hom_A\ a\ b] \otimes F_o\ a)$
using *comp-assoc* **by** *simp*
also have ... =
$Uncurry[G_o\ a,\ G_o\ b]\ (G_a\ a\ b) \cdot$
$(Hom_A\ a\ b \otimes \mathcal{T}\ a \cdot \mathrm{l}[F_o\ a]) \cdot$
$\mathrm{a}[Hom_A\ a\ b,\ \mathcal{I},\ F_o\ a] \cdot (\mathrm{r}^{-1}[Hom_A\ a\ b] \otimes F_o\ a)$
using *assms(4) [of a] a b F.preserves-Obj G.preserves-Obj UP-def Uncurry-Curry*
by *auto*
also have ... =
$Uncurry[G_o\ a,\ G_o\ b]\ (G_a\ a\ b) \cdot$
$((Hom_A\ a\ b \otimes \mathcal{T}\ a) \cdot (Hom_A\ a\ b \otimes \mathrm{l}[F_o\ a])) \cdot$
$\mathrm{a}[Hom_A\ a\ b,\ \mathcal{I},\ F_o\ a] \cdot (\mathrm{r}^{-1}[Hom_A\ a\ b] \otimes F_o\ a)$
using *assms(4) [of a] a b F.preserves-Obj G.preserves-Obj interchange*
by *auto*
also have ... =
$Uncurry[G_o\ a,\ G_o\ b]\ (G_a\ a\ b) \cdot (Hom_A\ a\ b \otimes \mathcal{T}\ a) \cdot$
$(Hom_A\ a\ b \otimes \mathrm{l}[F_o\ a]) \cdot \mathrm{a}[Hom_A\ a\ b,\ \mathcal{I},\ F_o\ a] \cdot$
$(\mathrm{r}^{-1}[Hom_A\ a\ b] \otimes F_o\ a)$
using *comp-assoc* **by** *simp*
also have ... = $Uncurry[G_o\ a,\ G_o\ b]\ (G_a\ a\ b) \cdot (Hom_A\ a\ b \otimes \mathcal{T}\ a)$
proof −
have $(Hom_A\ a\ b \otimes \mathrm{l}[F_o\ a]) \cdot \mathrm{a}[Hom_A\ a\ b,\ \mathcal{I},\ F_o\ a] \cdot$
$(\mathrm{r}^{-1}[Hom_A\ a\ b] \otimes F_o\ a) =$
$Hom_A\ a\ b \otimes F_o\ a$
proof −
have $(Hom_A\ a\ b \otimes \mathrm{l}[F_o\ a]) \cdot \mathrm{a}[Hom_A\ a\ b,\ \mathcal{I},\ F_o\ a] \cdot$
$(\mathrm{r}^{-1}[Hom_A\ a\ b] \otimes F_o\ a) =$
$(\mathrm{r}[Hom_A\ a\ b] \otimes F_o\ a) \cdot (\mathrm{r}^{-1}[Hom_A\ a\ b] \otimes F_o\ a)$
using *a b triangle* [*of* $Hom_A\ a\ b\ F_o\ a$]
by (*metis F.A.ide-Hom F.preserves-Obj comp-assoc mem-Collect-eq*)
also have ... = $\mathrm{r}[Hom_A\ a\ b] \cdot \mathrm{r}^{-1}[Hom_A\ a\ b] \otimes F_o\ a \cdot F_o\ a$
using *a b interchange F.preserves-Obj* **by** *force*
also have ... = $Hom_A\ a\ b \otimes F_o\ a$
using *a b F.preserves-Obj* **by** *auto*
finally show *?thesis* **by** *blast*

```
        qed
        thus ?thesis
          using assms(4) [of a] a b comp-arr-dom by auto
      qed
      also have ... = eval (G_o a) (G_o b) · (G_a a b ⊗ G_o a) · (Hom_A a b ⊗ 𝒯 a)
        using comp-assoc by auto
      also have ... = eval (G_o a) (G_o b) · (G_a a b ⊗ 𝒯 a)
        using assms(4) a b G.preserves-Hom comp-arr-dom comp-cod-arr
              interchange
        by (metis in-homE)
      finally have RHS: Uncurry[F_o a, G_o b]
                          (Comp (F_o a) (G_o a) (G_o b) · (G_a a b ⊗ (𝒯 a)↑) ·
                            r⁻¹[Hom_A a b]) =
                         eval (G_o a) (G_o b) · (G_a a b ⊗ 𝒯 a)
        by blast

      have Uncurry[F_o a, G_o b]
              (Comp (F_o a) (F_o b) (G_o b) · ((𝒯 b)↑ ⊗ F_a a b) · l⁻¹[Hom_A a b]) =
            Uncurry[F_o a, G_o b]
              (Comp (F_o a) (G_o a) (G_o b) · (G_a a b ⊗ (𝒯 a)↑) · r⁻¹[Hom_A a b])
        using a b assms(5) LHS RHS by simp
      moreover have «Comp (F_o a) (F_o b) (G_o b) ·
                       ((𝒯 b)↑ ⊗ F_a a b) · l⁻¹[Hom_A a b]
                       : Hom_A a b → exp (F_o a) (G_o b)»
          using assms(4) a b 1 F.preserves-Obj G.preserves-Obj
                F.preserves-Hom G.preserves-Hom
          apply (intro comp-in-homI' seqI)
                apply auto[1]
          by fastforce+
      moreover have «Comp (F_o a) (G_o a) (G_o b) ·
                       (G_a a b ⊗ (𝒯 a)↑) · r⁻¹[Hom_A a b]
                       : Hom_A a b → exp (F_o a) (G_o b)»
          using assms(4) a b 2 UP-DN(1) F.preserves-Obj G.preserves-Obj
                F.preserves-Hom G.preserves-Hom [of a b]
          apply (intro comp-in-homI' seqI)
                apply auto[7]
          by fastforce
      ultimately show Comp (F_o a) (F_o b) (G_o b) ·
                         ((𝒯 b)↑ ⊗ F_a a b) · l⁻¹[Hom_A a b] =
                        Comp (F_o a) (G_o a) (G_o b) ·
                         (G_a a b ⊗ (𝒯 a)↑) · r⁻¹[Hom_A a b]
        using a b Curry-Uncurry F.A.ide-Hom F.preserves-Obj
              G.preserves-Obj mem-Collect-eq
        by metis
    qed
  qed

end
```

Kelly (1.39) expresses enriched naturality in an alternate form, using the underlying functors of the covariant and contravariant enriched hom functors.

locale *Kelly-1-39* =
  symmetric-monoidal-category +
  elementary-closed-monoidal-category +
  enriched-natural-transformation
  for a :: $'a$
  and b :: $'a$ +
  assumes a: $a \in Obj_A$
  and b: $b \in Obj_A$
begin

  interpretation *enriched-category* $C\ T\ \alpha\ \iota$ ‹*Collect ide*› *exp Id Comp*
    using *is-enriched-in-itself* **by** *blast*
  interpretation *self-enriched-category* $C\ T\ \alpha\ \iota$ *exp eval Curry*
    ..

  sublocale *cov-Hom*: *covariant-Hom* $C\ T\ \alpha\ \iota$
      exp eval Curry Obj_B Hom_B Id_B $Comp_B$ ‹$F_o\ a$›
    using a *F.preserves-Obj* **by** *unfold-locales*
  sublocale *cnt-Hom*: *contravariant-Hom* $C\ T\ \alpha\ \iota\ \sigma$
      exp eval Curry Obj_B Hom_B Id_B $Comp_B$ ‹$G_o\ b$›
    using b *G.preserves-Obj* **by** *unfold-locales*

  lemma *Kelly-1-39*:
  shows $cov\text{-}Hom.map_0\ (F_o\ b)\ (G_o\ b)\ (\tau\ b) \cdot F_a\ a\ b =$
      $cnt\text{-}Hom.map_0\ (F_o\ a)\ (G_o\ a)\ (\tau\ a) \cdot G_a\ a\ b$
  proof −
    have $cov\text{-}Hom.map_0\ (F_o\ b)\ (G_o\ b)\ (\tau\ b) \cdot F_a\ a\ b =$
        $Comp_B\ (F_o\ a)\ (F_o\ b)\ (G_o\ b) \cdot (\tau\ b \otimes F_a\ a\ b) \cdot \mathrm{l}^{-1}[Hom_A\ a\ b]$
    proof −
      have $cov\text{-}Hom.map_0\ (F_o\ b)\ (G_o\ b)\ (\tau\ b) \cdot F_a\ a\ b =$
          $Comp_B\ (F_o\ a)\ (F_o\ b)\ (G_o\ b) \cdot$
            $(\tau\ b \otimes Hom_B\ (F_o\ a)\ (F_o\ b)) \cdot \mathrm{l}^{-1}[Hom_B\ (F_o\ a)\ (F_o\ b)] \cdot F_a\ a\ b$
        using *comp-assoc* **by** *simp*
      also have ... = $Comp_B\ (F_o\ a)\ (F_o\ b)\ (G_o\ b) \cdot$
          $(\tau\ b \otimes Hom_B\ (F_o\ a)\ (F_o\ b)) \cdot (\mathcal{I} \otimes F_a\ a\ b) \cdot \mathrm{l}^{-1}[Hom_A\ a\ b]$
        using a b *lunit'-naturality F.preserves-Hom* [*of* a b] **by** *fastforce*
      also have ... = $Comp_B\ (F_o\ a)\ (F_o\ b)\ (G_o\ b) \cdot$
          $((\tau\ b \otimes Hom_B\ (F_o\ a)\ (F_o\ b)) \cdot (\mathcal{I} \otimes F_a\ a\ b)) \cdot$
            $\mathrm{l}^{-1}[Hom_A\ a\ b]$
        using *comp-assoc* **by** *simp*
      also have ... = $Comp_B\ (F_o\ a)\ (F_o\ b)\ (G_o\ b) \cdot (\tau\ b \otimes F_a\ a\ b) \cdot$
          $\mathrm{l}^{-1}[Hom_A\ a\ b]$
        using a b *component-in-hom* [*of* b] *F.preserves-Hom* [*of* a b]
          comp-arr-dom comp-cod-arr [*of* $F_a\ a\ b\ Hom_B\ (F_o\ a)\ (F_o\ b)$]
          interchange
        by *fastforce*
      finally show *?thesis* **by** *blast*

qed
moreover have $\mathit{cnt\text{-}Hom.map_0}\ (F_o\ a)\ (G_o\ a)\ (\tau\ a) \cdot G_a\ a\ b =$
$\mathit{Comp_B}\ (F_o\ a)\ (G_o\ a)\ (G_o\ b) \cdot (G_a\ a\ b \otimes \tau\ a) \cdot \mathrm{r}^{-1}[\mathit{Hom_A}\ a\ b]$
proof –
have $\mathit{cnt\text{-}Hom.map_0}\ (F_o\ a)\ (G_o\ a)\ (\tau\ a) \cdot G_a\ a\ b =$
$\mathit{Comp_B}\ (F_o\ a)\ (G_o\ a)\ (G_o\ b) \cdot (\mathit{Hom_B}\ (G_o\ a)\ (G_o\ b) \otimes \tau\ a) \cdot$
$\mathrm{r}^{-1}[\mathit{Hom_B}\ (G_o\ a)\ (G_o\ b)] \cdot G_a\ a\ b$
using *comp-assoc* **by** *simp*
also have ... $= \mathit{Comp_B}\ (F_o\ a)\ (G_o\ a)\ (G_o\ b) \cdot$
$(\mathit{Hom_B}\ (G_o\ a)\ (G_o\ b) \otimes \tau\ a) \cdot (G_a\ a\ b \otimes \mathcal{I}) \cdot$
$\mathrm{r}^{-1}[\mathit{Hom_A}\ a\ b]$
using *a b runit′-naturality G.preserves-Hom* [*of a b*] **by** *fastforce*
also have ... $= \mathit{Comp_B}\ (F_o\ a)\ (G_o\ a)\ (G_o\ b) \cdot$
$((\mathit{Hom_B}\ (G_o\ a)\ (G_o\ b) \otimes \tau\ a) \cdot (G_a\ a\ b \otimes \mathcal{I})) \cdot$
$\mathrm{r}^{-1}[\mathit{Hom_A}\ a\ b]$
using *comp-assoc* **by** *simp*
also have ... $= \mathit{Comp_B}\ (F_o\ a)\ (G_o\ a)\ (G_o\ b) \cdot (G_a\ a\ b \otimes \tau\ a) \cdot$
$\mathrm{r}^{-1}[\mathit{Hom_A}\ a\ b]$
using *a b interchange component-in-hom* [*of a*] *G.preserves-Hom* [*of a b*]
comp-arr-dom comp-cod-arr [*of* G_a *a b* $\mathit{Hom_B}$ $(G_o\ a)\ (G_o\ b)$]
by *fastforce*
finally show *?thesis* **by** *blast*
qed
ultimately show *?thesis*
using *a b naturality* **by** *simp*
qed

end

2.5.2 Covariant Case

locale *covariant-yoneda-lemma* =
symmetric-monoidal-category +
C: *closed-symmetric-monoidal-category* +
covariant-Hom +
F: *enriched-functor C T* α ι *Obj Hom Id Comp* ‹*Collect ide*› *exp C.Id C.Comp*
begin

interpretation *C*: *elementary-closed-symmetric-monoidal-category C T* α ι σ *exp eval Curry* **..**
interpretation *C*: *self-enriched-category C T* α ι *exp eval Curry* **..**

Every element $e : \mathcal{I} \rightarrow F_o\ x$ of $F_o\ x$ determines an enriched natural transformation τ_e: $\mathit{hom}\ x - \rightarrow F$. The formula here is Kelly (1.47): $\tau_e\ y$: $\mathit{hom}\ x\ y \rightarrow F\ y$ is obtained as the composite:

$$\mathit{hom}\ x\ y \xrightarrow{F_a\ x\ y} \mathit{exp}\ (F\ x)\ (F\ y) \xrightarrow{\mathit{Exp}^{\leftarrow}\ e\ (F\ y)} \mathit{exp}\ \mathcal{I}\ (F\ y) \longrightarrow F\ y$$

where the third component is a canonical isomorphism. This basically amounts to evaluating $F_a\ x\ y$ on element e of $F_o\ x$ to obtain an element of

$F_o\ y$.

Note that the above composite gives an arrow $\tau_e\ y$: *hom x y* $\rightarrow$ $F\ y$, whereas the definition of enriched natural transformation formally requires $\tau_e\ y$: $\mathcal{I} \rightarrow$ *exp* (*hom x y*) ($F\ y$). So we need to transform the composite to achieve that.

```
abbreviation generated-transformation
where generated-transformation e ≡
      λy. (eval I (F_o y) · r⁻¹[exp I (F_o y)] · Exp← e (F_o y) · F_a x y)↑

lemma enriched-natural-transformation-generated-transformation:
assumes «e : I → F_o x»
shows enriched-natural-transformation C T α ι
        Obj Hom Id Comp (Collect ide) exp C.Id C.Comp
        hom_o hom_a F_o F_a (generated-transformation e)
proof (intro C.transformation-lam-UP)
  show ⋀y. y ∉ Obj ⟹
              eval I (F_o y) · r⁻¹[exp I (F_o y)] · Exp← e (F_o y) · F_a x y = null
    by (simp add: F.extensionality)
  show enriched-functor (·) T α ι Obj Hom Id Comp
        (Collect ide) exp C.Id C.Comp hom_o hom_a
    ..
  show enriched-functor (·) T α ι Obj Hom Id Comp
        (Collect ide) exp C.Id C.Comp F_o F_a
    ..
  show *: ⋀y. y ∈ Obj ⟹
               «eval I (F_o y) · r⁻¹[exp I (F_o y)] · Exp← e (F_o y) · F_a x y
                 : hom_o y → F_o y»
    using assms x F.preserves-Obj F.preserves-Hom
    apply (intro in-homI seqI)
          apply auto[6]
    by fastforce+
  show ⋀a b. ⟦a ∈ Obj; b ∈ Obj⟧ ⟹
              (eval I (F_o b) · r⁻¹[exp I (F_o b)] ·
                 Exp← e (F_o b) · F_a x b) ·
                   Uncurry[hom_o a, hom_o b] (hom_a a b) =
              eval (F_o a) (F_o b) ·
                (F_a a b ⊗ eval I (F_o a) · r⁻¹[exp I (F_o a)] ·
                   Exp← e (F_o a) · F_a x a)
  proof −
    fix a b
    assume a: a ∈ Obj and b: b ∈ Obj
    have (eval I (F_o b) · r⁻¹[exp I (F_o b)] ·
            Exp← e (F_o b) · F_a x b) · Uncurry[hom_o a, hom_o b] (hom_a a b) =
          eval (F_o x) (F_o b) · (F_a x b · Comp x a b ⊗ e) ·
            r⁻¹[Hom a b ⊗ hom_o a]
    proof −
      have (eval I (F_o b) · r⁻¹[exp I (F_o b)] ·
              Exp← e (F_o b) · F_a x b) · Uncurry[hom_o a, hom_o b] (hom_a a b) =
```

```
      eval 𝓘 (F_o b) ·
          (r⁻¹[exp 𝓘 (F_o b)] · Exp← e (F_o b) · F_a x b) ·
              Uncurry[hom_o a, hom_o b] (hom_a a b)
  using comp-assoc by simp
also have ... = eval 𝓘 (F_o b) ·
                (r⁻¹[exp 𝓘 (F_o b)] · Exp← e (F_o b) · F_a x b) ·
                  Comp x a b
  using a b x C.Uncurry-Curry [of - hom_o a hom_o b] Comp-in-hom
  by auto
also have ... = eval 𝓘 (F_o b) ·
                ((Exp← e (F_o b) · F_a x b ⊗ 𝓘) ·
                    r⁻¹[hom_o b]) · Comp x a b
proof −
  have «Exp← e (F_o b) · F_a x b : hom_o b → exp 𝓘 (F_o b)»
    using assms a b x F.preserves-Obj F.preserves-Hom [of x b] by force
  thus ?thesis
    using a b F.preserves-Obj F.preserves-Hom
          runit'-naturality [of Exp← e (F_o b) · F_a x b]
    by auto
qed
also have ... = eval 𝓘 (F_o b) ·
                (((Exp← e (F_o b) ⊗ 𝓘) · (F_a x b ⊗ 𝓘)) ·
                    r⁻¹[hom_o b]) ·
                  Comp x a b
  using assms a b x F.preserves-Obj F.preserves-Hom [of x b]
        interchange [of Exp← e (F_o b) F_a x b 𝓘 𝓘]
  by fastforce
also have ... = Uncurry[𝓘, F_o b] (Exp← e (F_o b)) · (F_a x b ⊗ 𝓘) ·
                r⁻¹[hom_o b] · Comp x a b
  using comp-assoc by simp
also have ... = (eval (F_o x) (F_o b) · (exp (F_o x) (F_o b) ⊗ e)) ·
                 (F_a x b ⊗ 𝓘) · r⁻¹[hom_o b] · Comp x a b
  using assms a b x F.preserves-Obj C.Uncurry-Curry by auto
also have ... = eval (F_o x) (F_o b) ·
                ((exp (F_o x) (F_o b) ⊗ e) · (F_a x b ⊗ 𝓘)) ·
                 r⁻¹[hom_o b] · Comp x a b
  using comp-assoc by simp
also have ... = eval (F_o x) (F_o b) · (F_a x b ⊗ e) · r⁻¹[hom_o b] ·
                Comp x a b
  using assms a b x F.preserves-Hom [of x b]
        comp-cod-arr [of F_a x b exp (F_o x) (F_o b)] comp-arr-dom
        interchange
  by fastforce
also have ... = eval (F_o x) (F_o b) · (F_a x b ⊗ e) ·
                (Comp x a b ⊗ 𝓘) · r⁻¹[Hom a b ⊗ hom_o a]
  using assms a b x runit'-naturality [of Comp x a b]
        Comp-in-hom [of x a b]
  by auto
also have ... = eval (F_o x) (F_o b) · ((F_a x b ⊗ e) · (Comp x a b ⊗ 𝓘)) ·
```

$\mathrm{r}^{-1}[\mathit{Hom}\ a\ b \otimes \mathit{hom}_o\ a]$
using *comp-assoc* **by** *simp*
also have ... $= \mathit{eval}\ (F_o\ x)\ (F_o\ b) \cdot (F_a\ x\ b \cdot \mathit{Comp}\ x\ a\ b \otimes e) \cdot$
$\mathrm{r}^{-1}[\mathit{Hom}\ a\ b \otimes \mathit{hom}_o\ a]$
using *assms a b x F.preserves-Hom* [*of x b*] *Comp-in-hom comp-arr-dom*
interchange [*of* $F_a\ x\ b$ *Comp x a b e* $\mathcal{I}$]
by *fastforce*
finally show *?thesis* **by** *blast*
qed
also have ... $= \mathit{eval}\ (F_o\ a)\ (F_o\ b) \cdot$
$(F_a\ a\ b \otimes \mathit{eval}\ \mathcal{I}\ (F_o\ a) \cdot \mathrm{r}^{-1}[\mathit{exp}\ \mathcal{I}\ (F_o\ a)] \cdot$
$\mathit{Exp}^{\leftarrow}\ e\ (F_o\ a) \cdot F_a\ x\ a)$
proof −
have $\mathit{eval}\ (F_o\ a)\ (F_o\ b) \cdot$
$(F_a\ a\ b \otimes \mathit{eval}\ \mathcal{I}\ (F_o\ a) \cdot \mathrm{r}^{-1}[\mathit{exp}\ \mathcal{I}\ (F_o\ a)] \cdot$
$\mathit{Exp}^{\leftarrow}\ e\ (F_o\ a) \cdot F_a\ x\ a) =$
$\mathit{eval}\ (F_o\ a)\ (F_o\ b) \cdot$
$(F_a\ a\ b \otimes \mathit{eval}\ \mathcal{I}\ (F_o\ a) \cdot (\mathrm{r}^{-1}[\mathit{exp}\ \mathcal{I}\ (F_o\ a)] \cdot$
$\mathit{Exp}^{\leftarrow}\ e\ (F_o\ a)) \cdot F_a\ x\ a)$
using *comp-assoc* **by** *simp*
also have ... $=$
$\mathit{eval}\ (F_o\ a)\ (F_o\ b) \cdot$
$(F_a\ a\ b \otimes \mathit{eval}\ \mathcal{I}\ (F_o\ a) \cdot$
$((\mathit{Exp}^{\leftarrow}\ e\ (F_o\ a) \otimes \mathcal{I}) \cdot \mathrm{r}^{-1}[\mathit{exp}\ (F_o\ x)\ (F_o\ a)]) \cdot$
$F_a\ x\ a)$
using *assms a b x F.preserves-Obj F.preserves-Hom*
runit′-naturality [*of* $\mathit{Exp}^{\leftarrow}\ e\ (F_o\ a)$]
by *auto*
also have ... $=$
$\mathit{eval}\ (F_o\ a)\ (F_o\ b) \cdot$
$(F_a\ a\ b \otimes$
$\mathit{Uncurry}[\mathcal{I},\ F_o\ a]\ (\mathit{Exp}^{\leftarrow}\ e\ (F_o\ a)) \cdot \mathrm{r}^{-1}[\mathit{exp}\ (F_o\ x)\ (F_o\ a)] \cdot$
$F_a\ x\ a)$
using *comp-assoc* **by** *simp*
also have ... $=$
$\mathit{eval}\ (F_o\ a)\ (F_o\ b) \cdot$
$(F_a\ a\ b \otimes$
$(\mathit{eval}\ (F_o\ x)\ (F_o\ a) \cdot (\mathit{exp}\ (F_o\ x)\ (F_o\ a) \otimes e)) \cdot$
$\mathrm{r}^{-1}[\mathit{exp}\ (F_o\ x)\ (F_o\ a)] \cdot F_a\ x\ a)$
using *assms a b x F.preserves-Obj C.Uncurry-Curry* **by** *auto*
also have ... $=$
$\mathit{eval}\ (F_o\ a)\ (F_o\ b) \cdot$
$(F_a\ a\ b \otimes$
$(\mathit{eval}\ (F_o\ x)\ (F_o\ a) \cdot (\mathit{exp}\ (F_o\ x)\ (F_o\ a) \otimes e)) \cdot$
$(F_a\ x\ a \otimes \mathcal{I}) \cdot \mathrm{r}^{-1}[\mathit{hom}_o\ a])$
using *a b x F.preserves-Hom* [*of x a*] *runit′-naturality* **by** *fastforce*
also have ... $=$
$\mathit{eval}\ (F_o\ a)\ (F_o\ b) \cdot$
$(F_a\ a\ b \otimes$

eval $(F_o\ x)$ $(F_o\ a)$ $\cdot$ (*exp* $(F_o\ x)$ $(F_o\ a) \otimes e) \cdot$
$(F_a\ x\ a \otimes \mathcal{I}) \cdot \mathrm{r}^{-1}[hom_o\ a])$
using *comp-assoc* **by** *simp*
also have ... =
eval $(F_o\ a)$ $(F_o\ b) \cdot$
$(F_a\ a\ b \otimes$ *eval* $(F_o\ x)$ $(F_o\ a) \cdot (F_a\ x\ a \otimes e) \cdot \mathrm{r}^{-1}[hom_o\ a])$
using *assms a b x F.preserves-Obj F.preserves-Hom F.preserves-Hom*
comp-arr-dom [*of e* $\mathcal{I}$]
comp-cod-arr [*of* F_a *x a exp* $(F_o\ x)$ $(F_o\ a)$]
interchange [*of exp* $(F_o\ x)$ $(F_o\ a)$ F_a *x a e* $\mathcal{I}$] *comp-assoc*
by (*metis in-homE*)
also have ... =
eval $(F_o\ a)$ $(F_o\ b) \cdot$
(*exp* $(F_o\ a)$ $(F_o\ b) \otimes$ *eval* $(F_o\ x)$ $(F_o\ a)) \cdot$
$(F_a\ a\ b \otimes (F_a\ x\ a \otimes e) \cdot \mathrm{r}^{-1}[hom_o\ a])$
using *assms a b x F.preserves-Obj F.preserves-Hom* [*of x a*]
F.preserves-Hom [*of a b*]
comp-cod-arr [*of* F_a *a b exp* $(F_o\ a)$ $(F_o\ b)$]
interchange
[*of exp* $(F_o\ a)$ $(F_o\ b)$ F_a *a b*
eval $(F_o\ x)$ $(F_o\ a)$ $(F_a\ x\ a \otimes e) \cdot \mathrm{r}^{-1}[hom_o\ a]$]
by *fastforce*
also have ... = (*eval* $(F_o\ a)$ $(F_o\ b) \cdot$
(*exp* $(F_o\ a)$ $(F_o\ b) \otimes$ *eval* $(F_o\ x)$ $(F_o\ a))) \cdot$
$(F_a\ a\ b \otimes (F_a\ x\ a \otimes e) \cdot \mathrm{r}^{-1}[hom_o\ a])$
using *comp-assoc* **by** *simp*
also have ... = (*eval* $(F_o\ a)$ $(F_o\ b) \cdot$
(*exp* $(F_o\ a)$ $(F_o\ b) \otimes$ *eval* $(F_o\ x)$ $(F_o\ a))) \cdot$
$(F_a\ a\ b \otimes (F_a\ x\ a \otimes e)) \cdot$ (*Hom a b* $\otimes\ \mathrm{r}^{-1}[hom_o\ a])$
using *assms a b x F.preserves-Obj F.preserves-Hom* [*of a b*]
F.preserves-Hom [*of x a*] *comp-arr-dom* [*of* F_a *a b Hom a b*]
interchange [*of* F_a *a b Hom a b* F_a *x a* $\otimes$ *e* $\mathrm{r}^{-1}[hom_o\ a]$]
by *fastforce*
also have ... = (*eval* $(F_o\ a)$ $(F_o\ b) \cdot$
(*exp* $(F_o\ a)$ $(F_o\ b) \otimes$ *eval* $(F_o\ x)$ $(F_o\ a))) \cdot$
(*exp* $(F_o\ a)$ $(F_o\ b) \otimes$ *exp* $(F_o\ x)$ $(F_o\ a) \otimes F_o\ x) \cdot$
$(F_a\ a\ b \otimes F_a\ x\ a \otimes e) \cdot$ (*Hom a b* $\otimes\ \mathrm{r}^{-1}[hom_o\ a])$
using *assms a b x F.preserves-Obj F.preserves-Hom* [*of a b*]
F.preserves-Hom [*of x a*]
comp-cod-arr [*of* $(F_a\ a\ b \otimes F_a\ x\ a \otimes e) \cdot$ (*Hom a b* $\otimes\ \mathrm{r}^{-1}[hom_o\ a])$
exp $(F_o\ a)$ $(F_o\ b) \otimes$ *exp* $(F_o\ x)$ $(F_o\ a) \otimes F_o\ x$]
by *fastforce*
also have ... = (*eval* $(F_o\ a)$ $(F_o\ b) \cdot$
(*exp* $(F_o\ a)$ $(F_o\ b) \otimes$ *eval* $(F_o\ x)$ $(F_o\ a))) \cdot$
(a[*exp* $(F_o\ a)$ $(F_o\ b)$, *exp* $(F_o\ x)$ $(F_o\ a)$, $F_o\ x$] $\cdot$
a^{-1}[*exp* $(F_o\ a)$ $(F_o\ b)$, *exp* $(F_o\ x)$ $(F_o\ a)$, $F_o\ x$]) $\cdot$
$(F_a\ a\ b \otimes (F_a\ x\ a \otimes e)) \cdot$ (*Hom a b* $\otimes\ \mathrm{r}^{-1}[hom_o\ a])$
using *assms a b x F.preserves-Obj comp-assoc-assoc'* **by** *simp*
also have ... = (*eval* $(F_o\ a)$ $(F_o\ b) \cdot$

$(exp\ (F_o\ a)\ (F_o\ b) \otimes eval\ (F_o\ x)\ (F_o\ a)) \cdot$
$\mathrm{a}[exp\ (F_o\ a)\ (F_o\ b),\ exp\ (F_o\ x)\ (F_o\ a),\ F_o\ x]) \cdot$
$(\mathrm{a}^{-1}[exp\ (F_o\ a)\ (F_o\ b),\ exp\ (F_o\ x)\ (F_o\ a),\ F_o\ x] \cdot$
$(F_a\ a\ b \otimes F_a\ x\ a \otimes e)) \cdot (Hom\ a\ b \otimes \mathrm{r}^{-1}[hom_o\ a])$
using *comp-assoc* **by** *simp*
also have ... $= Uncurry[F_o\ x,\ F_o\ b]\ (C.Comp\ (F_o\ x)\ (F_o\ a)\ (F_o\ b)) \cdot$
$(\mathrm{a}^{-1}[exp\ (F_o\ a)\ (F_o\ b),\ exp\ (F_o\ x)\ (F_o\ a),\ F_o\ x] \cdot$
$(F_a\ a\ b \otimes F_a\ x\ a \otimes e)) \cdot (Hom\ a\ b \otimes \mathrm{r}^{-1}[hom_o\ a])$
using *assms a b x F.preserves-Obj C.Uncurry-Curry C.Comp-def*
by *auto*
also have ... $= Uncurry[F_o\ x,\ F_o\ b]\ (C.Comp\ (F_o\ x)\ (F_o\ a)\ (F_o\ b)) \cdot$
$(((F_a\ a\ b \otimes F_a\ x\ a) \otimes e) \cdot \mathrm{a}^{-1}[Hom\ a\ b,\ hom_o\ a,\ \mathcal{I}]) \cdot$
$(Hom\ a\ b \otimes \mathrm{r}^{-1}[hom_o\ a])$
using *assms a b x F.preserves-Hom* [*of a b*] *F.preserves-Hom* [*of x a*]
assoc′-naturality [*of* $F_a\ a\ b\ F_a\ x\ a\ e$]
by *fastforce*
also have ... $= eval\ (F_o\ x)\ (F_o\ b) \cdot$
$((C.Comp\ (F_o\ x)\ (F_o\ a)\ (F_o\ b) \otimes F_o\ x) \cdot$
$((F_a\ a\ b \otimes F_a\ x\ a) \otimes e)) \cdot$
$\mathrm{a}^{-1}[Hom\ a\ b,\ hom_o\ a,\ \mathcal{I}] \cdot (Hom\ a\ b \otimes \mathrm{r}^{-1}[hom_o\ a])$
using *comp-assoc* **by** *simp*
also have ... $= eval\ (F_o\ x)\ (F_o\ b) \cdot$
$(C.Comp\ (F_o\ x)\ (F_o\ a)\ (F_o\ b) \cdot (F_a\ a\ b \otimes F_a\ x\ a) \otimes e) \cdot$
$\mathrm{a}^{-1}[Hom\ a\ b,\ hom_o\ a,\ \mathcal{I}] \cdot (Hom\ a\ b \otimes \mathrm{r}^{-1}[hom_o\ a])$
using *assms a b x F.preserves-Obj F.preserves-Hom* [*of a b*]
F.preserves-Hom [*of x a*] *comp-cod-arr* [*of e* $F_o\ x$]
interchange
[*of C.Comp* $(F_o\ x)\ (F_o\ a)\ (F_o\ b)\ F_a\ a\ b \otimes F_a\ x\ a\ F_o\ x\ e$]
by *fastforce*
also have ... $= eval\ (F_o\ x)\ (F_o\ b) \cdot (F_a\ x\ b \cdot Comp\ x\ a\ b \otimes e) \cdot$
$\mathrm{a}^{-1}[Hom\ a\ b,\ hom_o\ a,\ \mathcal{I}] \cdot (Hom\ a\ b \otimes \mathrm{r}^{-1}[hom_o\ a])$
using *assms a b x F.preserves-Obj F.preserves-Hom F.preserves-Comp*
by *simp*
also have ... $= eval\ (F_o\ x)\ (F_o\ b) \cdot (F_a\ x\ b \cdot Comp\ x\ a\ b \otimes e) \cdot$
$\mathrm{r}^{-1}[Hom\ a\ b \otimes hom_o\ a]$
proof −
have $\mathrm{a}^{-1}[Hom\ a\ b,\ hom_o\ a,\ \mathcal{I}] \cdot (Hom\ a\ b \otimes \mathrm{r}^{-1}[hom_o\ a]) =$
$\mathrm{r}^{-1}[Hom\ a\ b \otimes hom_o\ a]$
proof −
have $\mathrm{a}^{-1}[Hom\ a\ b,\ hom_o\ a,\ \mathcal{I}] \cdot (Hom\ a\ b \otimes \mathrm{r}^{-1}[hom_o\ a]) =$
$inv\ ((Hom\ a\ b \otimes \mathrm{r}[hom_o\ a]) \cdot \mathrm{a}[Hom\ a\ b,\ hom_o\ a,\ \mathcal{I}])$
using *assms a b x inv-comp* **by** *auto*
also have ... $= \mathrm{r}^{-1}[Hom\ a\ b \otimes hom_o\ a]$
using *assms a b x runit-tensor* **by** *auto*
finally show *?thesis* **by** *blast*
qed
thus *?thesis* **by** *simp*
qed
finally show *?thesis* **by** *simp*

qed
finally show $(eval\ \mathcal{I}\ (F_o\ b) \cdot \mathrm{r}^{-1}[exp\ \mathcal{I}\ (F_o\ b)] \cdot Exp^{\leftarrow}\ e\ (F_o\ b) \cdot F_a\ x\ b) \cdot$
$Uncurry[hom_o\ a,\ hom_o\ b]\ (hom_a\ a\ b) =$
$eval\ (F_o\ a)\ (F_o\ b) \cdot (F_a\ a\ b \otimes eval\ \mathcal{I}\ (F_o\ a) \cdot \mathrm{r}^{-1}[exp\ \mathcal{I}\ (F_o\ a)] \cdot$
$Exp^{\leftarrow}\ e\ (F_o\ a) \cdot F_a\ x\ a)$
by *argo*
qed
qed

If τ: *hom* $x - \rightarrow F$ is an enriched natural transformation, then there exists an element $e_\tau : \mathcal{I} \rightarrow F\ x$ that generates τ via the preceding formula. The idea (Kelly 1.46) is to take:

$$e_\tau = \mathcal{I} \xrightarrow{Id\ x} hom_o\ x \xrightarrow{\tau\ x} F\ x$$

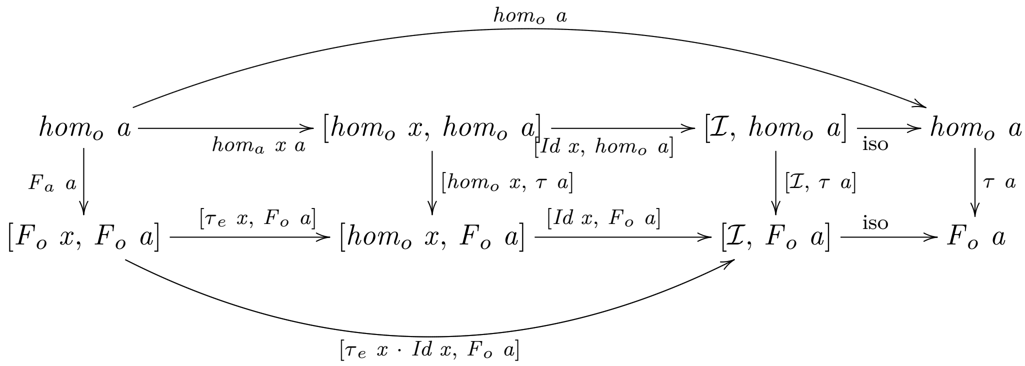
This amounts to the "evaluation of $\tau\ x$ at the identity on x".

However, note once again that, according to the formal definition of enriched natural transformation, we have $\tau\ x : \mathcal{I} \rightarrow exp\ (hom_o\ x)\ (F_o\ x)$, so it is necessary to transform this to an arrow: $(\tau\ x)\ {}^{\downarrow}[hom_o\ x,\ F_o\ x]$: hom_o $x \rightarrow F\ x$.

abbreviation *generating-elem*
where *generating-elem* $\tau \equiv (\tau\ x)\ {}^{\downarrow}[hom_o\ x,\ F_o\ x] \cdot Id\ x$

lemma *generating-elem-in-hom*:
assumes *enriched-natural-transformation C T* $\alpha\ \iota$
Obj Hom Id Comp (*Collect ide*) *exp C.Id C.Comp*
$hom_o\ hom_a\ F_o\ F_a\ \tau$
shows «*generating-elem* $\tau : \mathcal{I} \rightarrow F_o\ x$»
proof −
interpret τ: *enriched-natural-transformation C T* $\alpha\ \iota$
Obj Hom Id Comp ‹*Collect ide*› *exp C.Id C.Comp*
$hom_o\ hom_a\ F_o\ F_a\ \tau$
using *assms* **by** *blast*
show «*generating-elem* $\tau : \mathcal{I} \rightarrow F_o\ x$»
using *x Id-in-hom* τ*.component-in-hom* [*of x*] *F.preserves-Obj C.DN-def*
by *auto fastforce*
qed

Now we have to verify the elements of the diagram after Kelly (1.47):



The left square is enriched naturality of τ (Kelly (1.39)). The middle square commutes trivially. The right square commutes by the naturality of the canonical isomorphismm from $[\mathcal{I},\ hom_o\ a]$ to $hom_o\ a$. The top edge composes to $hom_o\ a$ (an identity). The commutativity of the entire diagram shows that $\tau\ a$ is recovered from e_τ. Note that where $\tau\ a$ appears, what is actually meant formally is $(\tau\ a)\ {\downarrow}[hom_o\ a,\ F_o\ a]$.

lemma *center-square*:
assumes *enriched-natural-transformation* $C\ T\ \alpha\ \iota$
Obj Hom Id Comp (*Collect ide*) *exp* $C.Id\ C.Comp$
$hom_o\ hom_a\ F_o\ F_a\ \tau$
and $a \in Obj$
shows $C.Exp\ \mathcal{I}\ (\tau\ a\ {\downarrow}[hom_o\ a,\ F_o\ a]) \cdot C.Exp\ (Id\ x)\ (hom_o\ a) =$
$C.Exp\ (Id\ x)\ (F_o\ a) \cdot C.Exp\ (hom_o\ x)\ (\tau\ a\ {\downarrow}[hom_o\ a,\ F_o\ a])$
proof −
interpret τ: *enriched-natural-transformation* $C\ T\ \alpha\ \iota$
Obj Hom Id Comp ‹*Collect ide*› *exp* $C.Id\ C.Comp$
$hom_o\ hom_a\ F_o\ F_a\ \tau$
using *assms* **by** *blast*
let $?\tau_a = \tau\ a\ {\downarrow}[hom_o\ a,\ F_o\ a]$
show $C.Exp\ \mathcal{I}\ ?\tau_a \cdot C.Exp\ (Id\ x)\ (hom_o\ a) =$
$C.Exp\ (Id\ x)\ (F_o\ a) \cdot C.Exp\ (hom_o\ x)\ ?\tau_a$
by (*metis assms*(*2*) *x C.Exp-comp F.preserves-Obj Id-in-hom*
C.DN-simps(*1*−*3*) *comp-arr-dom comp-cod-arr in-homE* τ*.component-in-hom*
ide-Hom mem-Collect-eq)
qed

lemma *right-square*:
assumes *enriched-natural-transformation* $C\ T\ \alpha\ \iota$
Obj Hom Id Comp (*Collect ide*) *exp* $C.Id\ C.Comp$

```
        homo homa Fo Fa τ
and a ∈ Obj
shows τ a ↓[homo a, Fo a] · C.Dn (homo a) =
      C.Dn (Fo a) · C.Exp ℐ (τ a ↓[homo a, Fo a])
proof –
  interpret τ: enriched-natural-transformation C T α ι
             Obj Hom Id Comp ‹Collect ide› exp C.Id C.Comp
             homo homa Fo Fa τ
    using assms by blast
  show ?thesis
    using assms(2) C.Up-Dn-naturality C.DN-simps τ.component-in-hom
    apply auto[1]
    by (metis C.Exp-ide-y C.UP-DN(2) F.preserves-Obj ide-Hom ide-unity
        in-homE mem-Collect-eq x)
qed

lemma top-path:
assumes a ∈ Obj
shows eval ℐ (homo a) · r⁻¹[exp ℐ (homo a)] · C.Exp (Id x) (homo a) ·
        homa x a =
      homo a
proof –
  have eval ℐ (homo a) · r⁻¹[exp ℐ (homo a)] · C.Exp (Id x) (homo a) ·
         homa x a =
       eval ℐ (homo a) · r⁻¹[exp ℐ (homo a)] ·
         (Curry[exp ℐ (homo a), ℐ, homo a] (homo a · eval ℐ (homo a)) ·
            Curry[exp (homo x) (homo a), ℐ, homo a]
              (eval (homo x) (homo a) · (exp (homo x) (homo a) ⊗ Id x))) ·
         homa x a
    using assms x C.Exp-def Id-in-hom [of x] by auto
  also have ... =
       eval ℐ (homo a) · r⁻¹[exp ℐ (homo a)] ·
         Curry[exp ℐ (homo a), ℐ, homo a] (homo a · eval ℐ (homo a)) ·
           Curry[exp (homo x) (homo a), ℐ, homo a]
             (eval (homo x) (homo a) · (exp (homo x) (homo a) ⊗ Id x)) ·
               homa x a
    using comp-assoc by simp
  also have ... =
       eval ℐ (homo a) · r⁻¹[exp ℐ (homo a)] ·
         Curry[exp ℐ (homo a), ℐ, homo a] (homo a · eval ℐ (homo a)) ·
           Curry[homo a, ℐ, homo a]
             ((eval (homo x) (homo a) · (exp (homo x) (homo a) ⊗ Id x)) ·
               (homa x a ⊗ ℐ))
  proof –
    have «eval (homo x) (homo a) · (exp (homo x) (homo a) ⊗ Id x)
             : exp (homo x) (homo a) ⊗ ℐ → homo a»
      using assms x
      by (meson Id-in-hom comp-in-homI C.eval-in-hom-ax C.ide-exp
          ide-in-hom tensor-in-hom ide-Hom)
```

thus *?thesis*
  using *assms x preserves-Hom* [*of x a*] *C.comp-Curry-arr* **by** *simp*
qed
also have ... =
  $eval\ \mathcal{I}\ (hom_o\ a) \cdot r^{-1}[exp\ \mathcal{I}\ (hom_o\ a)] \cdot$
  $Curry[exp\ \mathcal{I}\ (hom_o\ a), \mathcal{I}, hom_o\ a]\ (hom_o\ a \cdot eval\ \mathcal{I}\ (hom_o\ a)) \cdot$
  $Curry[hom_o\ a, \mathcal{I}, hom_o\ a]$
  $(eval\ (hom_o\ x)\ (hom_o\ a) \cdot$
  $(exp\ (hom_o\ x)\ (hom_o\ a) \otimes Id\ x) \cdot (hom_a\ x\ a \otimes \mathcal{I}))$
using *comp-assoc* **by** *simp*
also have ... =
  $eval\ \mathcal{I}\ (hom_o\ a) \cdot r^{-1}[exp\ \mathcal{I}\ (hom_o\ a)] \cdot$
  $Curry[exp\ \mathcal{I}\ (hom_o\ a), \mathcal{I}, hom_o\ a]\ (hom_o\ a \cdot eval\ \mathcal{I}\ (hom_o\ a)) \cdot$
  $Curry[hom_o\ a, \mathcal{I}, hom_o\ a]$
  $(eval\ (hom_o\ x)\ (hom_o\ a) \cdot (hom_a\ x\ a \otimes Id\ x))$
proof −
have $seq\ (Id\ x)\ \mathcal{I} \wedge seq\ (hom_o\ x)\ (Id\ x)$
  using *x Id-in-hom ide-in-hom ide-unity* **by** *blast*
thus *?thesis*
  using *assms x preserves-Hom comp-arr-dom* [*of Id x* $\mathcal{I}$]
  interchange [*of exp* $(hom_o\ x)\ (hom_o\ a)\ hom_a\ x\ a\ Id\ x\ \mathcal{I}$]
  by (*metis comp-cod-arr comp-ide-arr dom-eqI ide-unity in-homE ide-Hom*)
qed
also have ... =
  $eval\ \mathcal{I}\ (hom_o\ a) \cdot r^{-1}[exp\ \mathcal{I}\ (hom_o\ a)] \cdot$
  $Curry[exp\ \mathcal{I}\ (hom_o\ a), \mathcal{I}, hom_o\ a]\ (eval\ \mathcal{I}\ (hom_o\ a)) \cdot$
  $Curry[hom_o\ a, \mathcal{I}, hom_o\ a]$
  $(Uncurry[hom_o\ x, hom_o\ a]\ (hom_a\ x\ a) \cdot (hom_o\ a \otimes Id\ x))$
proof −
have $eval\ (hom_o\ x)\ (hom_o\ a) \cdot (hom_a\ x\ a \otimes Id\ x) =$
  $eval\ (hom_o\ x)\ (hom_o\ a) \cdot (hom_a\ x\ a \cdot hom_o\ a \otimes hom_o\ x \cdot Id\ x)$
  using *assms x Id-in-hom comp-cod-arr comp-arr-dom Comp-in-hom*
  by (*metis in-homE preserves-Hom*)
also have ... = $eval\ (hom_o\ x)\ (hom_o\ a) \cdot (hom_a\ x\ a \otimes hom_o\ x) \cdot$
  $(hom_o\ a \otimes Id\ x)$
  using *assms x Id-in-hom Comp-in-hom*
  interchange [*of* $hom_a\ x\ a\ hom_o\ a\ hom_o\ x\ Id\ x$]
  by (*metis comp-arr-dom comp-cod-arr in-homE preserves-Hom*)
also have ... = $Uncurry[hom_o\ x, hom_o\ a]\ (hom_a\ x\ a) \cdot (hom_o\ a \otimes Id\ x)$
  using *comp-assoc* **by** *simp*
finally show *?thesis*
  using *assms x comp-cod-arr ide-Hom ide-unity C.eval-simps*(*1*,*3*) **by** *metis*
qed
also have ... =
  $eval\ \mathcal{I}\ (hom_o\ a) \cdot r^{-1}[exp\ \mathcal{I}\ (hom_o\ a)] \cdot$
  $Curry[hom_o\ a, \mathcal{I}, hom_o\ a]$
  $(Uncurry[\mathcal{I}, hom_o\ a]$
  $(Curry[hom_o\ a, \mathcal{I}, hom_o\ a]$

$(Uncurry[hom_o\ x,\ hom_o\ a]\ (hom_a\ x\ a) \cdot (hom_o\ a \otimes Id\ x))))$
using *assms x*
C.comp-Curry-arr
[of $\mathcal{I}$
Curry[hom_o a, $\mathcal{I}$, hom_o a]
$(Uncurry[hom_o\ x,\ hom_o\ a]\ (hom_a\ x\ a) \cdot (hom_o\ a \otimes Id\ x))$
hom_o a exp $\mathcal{I}$ (hom_o a)
eval $\mathcal{I}$ (hom_o a) hom_o a]
apply *auto[1]*
by (*metis Comp-Hom-Id Comp-in-hom C.Uncurry-Curry C.eval-in-hom-ax ide-unity C.isomorphic-exp-unity(1) ide-Hom*)
also have ... =
$eval\ \mathcal{I}\ (hom_o\ a) \cdot \mathrm{r}^{-1}[exp\ \mathcal{I}\ (hom_o\ a)] \cdot$
$Curry[hom_o\ a,\ \mathcal{I},\ hom_o\ a]$
$(Uncurry[hom_o\ x,\ hom_o\ a]\ (hom_a\ x\ a) \cdot (hom_o\ a \otimes Id\ x))$
using *assms x C.Uncurry-Curry*
by (*simp add*: *Comp-Hom-Id Comp-in-hom C.Curry-Uncurry C.isomorphic-exp-unity(1)*)
also have ... =
$eval\ \mathcal{I}\ (hom_o\ a) \cdot \mathrm{r}^{-1}[exp\ \mathcal{I}\ (hom_o\ a)] \cdot$
$Curry[hom_o\ a,\ \mathcal{I},\ hom_o\ a]$
$(eval\ (hom_o\ x)\ (hom_o\ a) \cdot (hom_a\ x\ a \otimes hom_o\ x) \cdot (hom_o\ a \otimes Id\ x))$
using *comp-assoc* **by** *simp*
also have ... =
$eval\ \mathcal{I}\ (hom_o\ a) \cdot \mathrm{r}^{-1}[exp\ \mathcal{I}\ (hom_o\ a)] \cdot$
$Curry[hom_o\ a,\ \mathcal{I},\ hom_o\ a]$
$(eval\ (hom_o\ x)\ (hom_o\ a) \cdot (hom_a\ x\ a \otimes Id\ x))$
using *assms x comp-cod-arr [of Id x hom_o x] comp-arr-dom*
interchange [of hom_a x a hom_o a hom_o x Id x]
preserves-Hom [of x a] Id-in-hom
apply *auto[1]*
by *fastforce*
also have ... =
$eval\ \mathcal{I}\ (hom_o\ a) \cdot$
$(Curry[hom_o\ a,\ \mathcal{I},\ hom_o\ a]$
$(eval\ (hom_o\ x)\ (hom_o\ a) \cdot (hom_a\ x\ a \otimes Id\ x)) \otimes \mathcal{I}) \cdot$
$\mathrm{r}^{-1}[hom_o\ a]$
proof −
have «$Curry[hom_o\ a,\ \mathcal{I},\ hom_o\ a]$
$(eval\ (hom_o\ x)\ (hom_o\ a) \cdot (hom_a\ x\ a \otimes Id\ x))$
$: hom_o\ a \rightarrow exp\ \mathcal{I}\ (hom_o\ a)$»
using *assms x preserves-Hom [of x a] Id-in-hom [of x]* **by** *force*
thus *?thesis*
using *assms x runit′-naturality* **by** *fastforce*
qed
also have ... =
$Uncurry[\mathcal{I},\ hom_o\ a]$
$(Curry[hom_o\ a,\ \mathcal{I},\ hom_o\ a]$
$(eval\ (hom_o\ x)\ (hom_o\ a) \cdot (hom_a\ x\ a \otimes Id\ x))) \cdot \mathrm{r}^{-1}[hom_o\ a]$

using *comp-assoc* **by** *simp*
also have ... = (*eval* (hom_o *x*) (hom_o *a*) ·
(*Curry*[hom_o *a*, hom_o *x*, hom_o *a*] (*Comp x x a*) ⊗ *Id x*)) ·
r^{-1}[hom_o *a*]
using *assms x C.Uncurry-Curry preserves-Hom* [*of x a*] *Id-in-hom* [*of x*]
by *fastforce*
also have ... = (*eval* (hom_o *x*) (hom_o *a*) ·
((*Curry*[hom_o *a*, hom_o *x*, hom_o *a*] (*Comp x x a*) ⊗ hom_o *x*) ·
(hom_o *a* ⊗ *Id x*))) · r^{-1}[hom_o *a*]
using *assms x Id-in-hom* [*of x*] *Comp-in-hom comp-arr-dom comp-cod-arr interchange*
by *auto*
also have ... = *Uncurry*[hom_o *x*, hom_o *a*]
(*Curry*[hom_o *a*, hom_o *x*, hom_o *a*] (*Comp x x a*)) ·
(hom_o *a* ⊗ *Id x*) · r^{-1}[hom_o *a*]
using *comp-assoc* **by** *simp*
also have ... = *Comp x x a* · (hom_o *a* ⊗ *Id x*) · r^{-1}[hom_o *a*]
using *assms x C.Uncurry-Curry Comp-in-hom* **by** *simp*
also have ... = (*Comp x x a* · (hom_o *a* ⊗ *Id x*)) · r^{-1}[hom_o *a*]
using *comp-assoc* **by** *simp*
also have ... = r[hom_o *a*] · r^{-1}[hom_o *a*]
using *assms x Comp-Hom-Id* **by** *auto*
also have ... = hom_o *a*
using *assms x comp-runit-runit′* **by** *blast*
finally show *?thesis* **by** *blast*
qed

The left square is an instance of Kelly (1.39), so we can get that by instantiating that result. The confusing business is that the target enriched category is the base category C.

lemma *left-square*:
assumes *enriched-natural-transformation C T α ι*
Obj Hom Id Comp (*Collect ide*) *exp C.Id C.Comp*
hom_o hom_a F_o F_a τ
and *a* ∈ *Obj*
shows $Exp^{\rightarrow}$ (hom_o *x*) ((τ *a*) $^{\downarrow}$[hom_o *a*, F_o *a*]) · hom_a *x a* =
$Exp^{\leftarrow}$ ((τ *x*) $^{\downarrow}$[hom_o *x*, F_o *x*]) (F_o *a*) · F_a *x a*
proof −
interpret τ: *enriched-natural-transformation C T α ι*
Obj Hom Id Comp ‹*Collect ide*› *exp C.Id C.Comp*
hom_o hom_a F_o F_a τ
using *assms*(*1*) **by** *blast*

interpret *cov-Hom*: *covariant-Hom C T α ι exp eval Curry*
‹*Collect ide*› *exp C.Id C.Comp* ‹hom_o *x*›
using *x* **by** *unfold-locales auto*
interpret *cnt-Hom*: *contravariant-Hom C T α ι σ exp eval Curry*
‹*Collect ide*› *exp C.Id C.Comp* ‹F_o *a*›
using *assms*(*2*) *F.preserves-Obj* **by** *unfold-locales*

interpret *Kelly*: *Kelly-1-39* C T α ι σ *exp eval Curry*
Obj Hom Id Comp ‹*Collect ide*› *exp C.Id C.Comp*
hom_o hom_a F_o F_a τ x a
using *assms(2) x*
by *unfold-locales*

The following is the enriched naturality of τ, expressed in the alternate form involving the underlying ordinary functors of the enriched hom functors.

have *1*: *cov-Hom.map*$_0$ (hom_o a) (F_o a) (τ a) $\cdot$ hom_a x a =
cnt-Hom.map$_0$ (hom_o x) (F_o x) (τ x) $\cdot$ F_a x a
using *Kelly.Kelly-1-39* **by** *simp*

Here we have the underlying ordinary functor of the enriched covariant hom, expressed in terms of the covariant endofunctor $Exp^{\rightarrow}$ (hom_o x) on the base category.

have *2*: *cov-Hom.map*$_0$ (hom_o a) (F_o a) (τ a) =
$Exp^{\rightarrow}$ (hom_o x) ((τ a) $^{\downarrow}$[hom_o a, F_o a])
proof −
have *cov-Hom.map*$_0$ (hom_o a) (F_o a) (τ a) =
(*Curry*[*cnt-Hom.hom*$_o$ (hom_o a), *cov-Hom.hom*$_o$ (hom_o a),
cnt-Hom.hom$_o$ (hom_o x)]
(*C.Comp* (hom_o x) (hom_o a) (F_o a)) $\cdot$ τ a)
$^{\downarrow}$[*cov-Hom.hom*$_o$ (hom_o a), *cnt-Hom.hom*$_o$ (hom_o x)]
proof −
have *cov-Hom.map*$_0$ (hom_o a) (F_o a) (τ a) =
cnt-Hom.Op$_0$*.Map*
(*cov-Hom.UF.map*$_0$ (*cnt-Hom.Op*$_0$*.MkArr* (hom_o a) (F_o a) (τ a)))
$^{\downarrow}$[*cnt-Hom.Op*$_0$*.Dom*
(*cov-Hom.UF.map*$_0$
(*cnt-Hom.Op*$_0$*.MkArr* (hom_o a) (F_o a) (τ a))),
cnt-Hom.Op$_0$*.Cod*
(*cov-Hom.UF.map*$_0$
(*cnt-Hom.Op*$_0$*.MkArr* (hom_o a) (F_o a) (τ a)))]
using *assms x preserves-Obj F.preserves-Obj* τ*.component-in-hom*
cov-Hom.Kelly-1-31 cov-Hom.UF.preserves-arr
by *force*
moreover
have *cnt-Hom.Op*$_0$*.Dom*
(*cov-Hom.UF.map*$_0$
(*cnt-Hom.Op*$_0$*.MkArr* (hom_o a) (F_o a) (τ a))) =
exp (hom_o x) (hom_o a)
using *assms x cov-Hom.UF.map*$_0$*-def*
apply *auto[1]*
using *cnt-Hom.y* τ*.component-in-hom* **by** *force*
moreover
have *cnt-Hom.Op*$_0$*.Cod*
(*cov-Hom.UF.map*$_0$
(*cnt-Hom.Op*$_0$*.MkArr* (hom_o a) (F_o a) (τ a))) =

exp (hom_o *x*) (F_o *a*)
using *assms x cov-Hom.UF.map*$_0$*-def*
apply *auto*[*1*]
using *cnt-Hom.y* τ*.component-in-hom* **by** *fastforce*
moreover
have *cnt-Hom.Op*$_0$*.Map*
(*cov-Hom.UF.map*$_0$
(*cnt-Hom.Op*$_0$*.MkArr* (hom_o *a*) (F_o *a*) (τ *a*))) =
cov-Hom.hom$_a$ (hom_o *a*) (F_o *a*) $\cdot$ τ *a*
using *assms x cov-Hom.UF.map*$_0$*-def*
apply *auto*[*1*]
using *cnt-Hom.y* τ*.component-in-hom* **by** *auto*
ultimately show *?thesis*
using *assms x ide-Hom F.preserves-Obj* **by** *simp*
qed
also have ... = $Exp^{\rightarrow}$ (hom_o *x*) ((τ *a*) $^{\downarrow}$[hom_o *a*, F_o *a*])
using *assms*(*2*) *x C.cov-Exp-DN* τ*.component-in-hom F.preserves-Obj*
by *simp*
finally show *?thesis* **by** *blast*
qed

Here we have the underlying ordinary functor of the enriched contravariant hom, expressed in terms of the contravariant endofunctor $\lambda f.\ Exp^{\leftarrow}\ f$ $(F_o\ a)$ on the base category.

have *3*: *cnt-Hom.map*$_0$ (hom_o *x*) (F_o *x*) (τ *x*) =
$Exp^{\leftarrow}$ (τ *x* $^{\downarrow}$[hom_o *x*, F_o *x*]) (F_o *a*)
proof −
have *cnt-Hom.map*$_0$ (hom_o *x*) (F_o *x*) (τ *x*) =
Uncurry[*exp* (F_o *x*) (F_o *a*), *exp* (hom_o *x*) (F_o *a*)]
(*cnt-Hom.hom*$_a$ (F_o *x*) (hom_o *x*) $\cdot$ τ *x*) $\cdot$
l^{-1}[*exp* (F_o *x*) (F_o *a*)]
proof −
have *cnt-Hom.map*$_0$ (hom_o *x*) (F_o *x*) (τ *x*) =
Uncurry[*cnt-Hom.Op*$_0$*.Dom*
(*cnt-Hom.UF.map*$_0$
(*cnt-Hom.Op*$_0$*.MkArr* (F_o *x*) (hom_o *x*) (τ *x*))),
cnt-Hom.Op$_0$*.Cod*
(*cnt-Hom.UF.map*$_0$
(*cnt-Hom.Op*$_0$*.MkArr* (F_o *x*) (hom_o *x*) (τ *x*)))]
(*cnt-Hom.Op*$_0$*.Map*
(*cnt-Hom.UF.map*$_0$
(*cnt-Hom.Op*$_0$*.MkArr* (F_o *x*) (*Hom x x*) (τ *x*)))) $\cdot$
l^{-1}[*cnt-Hom.Op*$_0$*.Dom*
(*cnt-Hom.UF.map*$_0$
(*cnt-Hom.Op*$_0$*.MkArr* (F_o *x*) (hom_o *x*) (τ *x*)))]
using *assms x 1 2 cnt-Hom.Kelly-1-32* [*of* hom_o *x* F_o *x* τ *x*]
C.Curry-simps(*1*−*3*) *C.DN-def C.UP-DN*(*2*) *C.eval-simps*(*1*−*3*)
C.ide-exp Comp-in-hom F.preserves-Obj comp-in-homI'
not-arr-null preserves-Obj τ*.component-in-hom in-homE*

mem-Collect-eq seqE
by (*smt* (*verit*))
moreover have *cnt-Hom.*Op_0*.Dom*
(*cnt-Hom.UF.*map_0
(*cnt-Hom.*Op_0*.MkArr* (F_o *x*) (hom_o *x*) (τ *x*))) =
exp (F_o *x*) (F_o *a*)
using *assms x cnt-Hom.UF.*map_0*-def*
apply *auto*[*1*]
using *F.preserves-Obj cnt-Hom.*Op_0*.arr-MkArr* τ*.component-in-hom*
by *blast*
moreover have *cnt-Hom.*Op_0*.Cod*
(*cnt-Hom.UF.*map_0
(*cnt-Hom.*Op_0*.MkArr* (F_o *x*) (hom_o *x*) (τ *x*))) =
exp (hom_o *x*) (F_o *a*)
using *assms x cnt-Hom.UF.*map_0*-def*
apply *auto*[*1*]
using *F.preserves-Obj cnt-Hom.*Op_0*.arr-MkArr* τ*.component-in-hom*
by *blast*
moreover have *cnt-Hom.*Op_0*.Map*
(*cnt-Hom.UF.*map_0
(*cnt-Hom.*Op_0*.MkArr* (F_o *x*) (hom_o *x*) (τ *x*))) =
cnt-Hom.hom_a (F_o *x*) (hom_o *x*) · τ *x*
using *assms x cnt-Hom.UF.*map_0*-def F.preserves-Obj*
by (*simp add*: τ*.component-in-hom*)
ultimately show *?thesis* **by** *argo*
qed
also have ... = $Exp^{\leftarrow}$ (τ *x* $^{\downarrow}$[hom_o *x*, F_o *x*]) (F_o *a*)
using *assms*(*2*) *x* τ*.component-in-hom* [*of x*] *F.preserves-Obj*
C.DN-def C.cnt-Exp-DN
by *fastforce*
finally show *?thesis* **by** *simp*
qed
show *?thesis*
using *1 2 3* **by** *auto*
qed

lemma *transformation-generated-by-element*:
assumes *enriched-natural-transformation C T* α ι
Obj Hom Id Comp (*Collect ide*) *exp C.Id C.Comp*
hom_o hom_a F_o F_a τ
and *a* ∈ *Obj*
shows τ *a* = *generated-transformation* (*generating-elem* τ) *a*
proof −
interpret τ: *enriched-natural-transformation C T* α ι
Obj Hom Id Comp ‹*Collect ide*› *exp C.Id C.Comp*
hom_o hom_a F_o F_a τ
using *assms*(*1*) **by** *blast*
have τ *a* $^{\downarrow}$[hom_o *a*, F_o *a*] =
τ *a* $^{\downarrow}$[hom_o *a*, F_o *a*] ·

eval $\mathcal{I}$ (hom_o *a*) · r^{-1}[*exp* $\mathcal{I}$ (hom_o *a*)] · *C.Exp* (*Id x*) (hom_o *a*) ·
Curry[hom_o *a*, hom_o *x*, hom_o *a*] (*Comp x x a*)
using *assms(2) x top-path* τ*.component-in-hom* [*of a*] *F.preserves-Obj*
comp-arr-dom C.UP-DN(2)
by *auto*
also have ... =
(τ *a* $^{\downarrow}$[hom_o *a*, F_o *a*] · *eval* $\mathcal{I}$ (hom_o *a*) · r^{-1}[*exp* $\mathcal{I}$ (hom_o *a*)]) ·
C.Exp (*Id x*) (hom_o *a*) ·
Curry[hom_o *a*, hom_o *x*, hom_o *a*] (*Comp x x a*)
using *comp-assoc* **by** *simp*
also have ... =
(*eval* $\mathcal{I}$ (F_o *a*) · r^{-1}[*exp* $\mathcal{I}$ (F_o *a*)] ·
C.Exp $\mathcal{I}$ (*Uncurry*[hom_o *a*, F_o *a*] (τ *a*) · l^{-1}[hom_o *a*])) ·
C.Exp (*Id x*) (hom_o *a*) ·
Curry[hom_o *a*, hom_o *x*, hom_o *a*] (*Comp x x a*)
using *assms right-square C.DN-def* τ*.component-in-hom comp-assoc*
by *auto blast*
also have ... =
eval $\mathcal{I}$ (F_o *a*) · r^{-1}[*exp* $\mathcal{I}$ (F_o *a*)] ·
(*C.Exp* $\mathcal{I}$ (*Uncurry*[hom_o *a*, F_o *a*] (τ *a*) · l^{-1}[hom_o *a*]) ·
C.Exp (*Id x*) (hom_o *a*)) ·
Curry[hom_o *a*, hom_o *x*, hom_o *a*] (*Comp x x a*)
using *comp-assoc* **by** *simp*
also have ... =
eval $\mathcal{I}$ (F_o *a*) · r^{-1}[*exp* $\mathcal{I}$ (F_o *a*)] ·
(*C.Exp* (*Id x*) (F_o *a*) ·
C.Exp (hom_o *x*) (*Uncurry*[hom_o *a*, F_o *a*] (τ *a*) · l^{-1}[hom_o *a*])) ·
Curry[hom_o *a*, hom_o *x*, hom_o *a*] (*Comp x x a*)
using *assms center-square C.DN-def*
enriched-natural-transformation.component-in-hom
by *fastforce*
also have ... =
eval $\mathcal{I}$ (F_o *a*) · r^{-1}[*exp* $\mathcal{I}$ (F_o *a*)] · *C.Exp* (*Id x*) (F_o *a*) ·
C.Exp (hom_o *x*) (*Uncurry*[hom_o *a*, F_o *a*] (τ *a*) · l^{-1}[hom_o *a*]) ·
Curry[hom_o *a*, hom_o *x*, hom_o *a*] (*Comp x x a*)
using *comp-assoc* **by** *simp*
also have ... =
eval $\mathcal{I}$ (F_o *a*) · r^{-1}[*exp* $\mathcal{I}$ (F_o *a*)] · *C.Exp* (*Id x*) (F_o *a*) ·
$Exp^{\leftarrow}$ (*Uncurry*[hom_o *x*, F_o *x*] (τ *x*) · l^{-1}[hom_o *x*]) (F_o *a*) · F_a *x a*
proof −
have *eval* $\mathcal{I}$ (F_o *a*) · r^{-1}[*exp* $\mathcal{I}$ (F_o *a*)] · *C.Exp* (*Id x*) (F_o *a*) ·
C.Exp (hom_o *x*) (*Uncurry*[hom_o *a*, F_o *a*] (τ *a*) · l^{-1}[hom_o *a*]) ·
Curry[hom_o *a*, hom_o *x*, hom_o *a*] (*Comp x x a*) =
eval $\mathcal{I}$ (F_o *a*) · r^{-1}[*exp* $\mathcal{I}$ (F_o *a*)] · *C.Exp* (*Id x*) (F_o *a*) ·
C.Exp (hom_o *x*) (*Uncurry*[hom_o *a*, F_o *a*] (τ *a*) · l^{-1}[hom_o *a*]) ·
hom_a *x a*
using *assms(2) x* **by** *force*
also have ... =
eval $\mathcal{I}$ (F_o *a*) · r^{-1}[*exp* $\mathcal{I}$ (F_o *a*)] · *C.Exp* (*Id x*) (F_o *a*) ·

$Exp^{\rightarrow}\ (hom_o\ x)\ (Uncurry[hom_o\ a,\ F_o\ a]\ (\tau\ a) \cdot \mathrm{l}^{-1}[hom_o\ a]) \cdot$
$hom_a\ x\ a$
using *assms x C.Exp-def C.cnt-Exp-ide comp-arr-dom* **by** *auto*
also have ... =
$eval\ \mathcal{I}\ (F_o\ a) \cdot \mathrm{r}^{-1}[exp\ \mathcal{I}\ (F_o\ a)] \cdot C.Exp\ (Id\ x)\ (F_o\ a) \cdot$
$Exp^{\rightarrow}\ (hom_o\ x)\ (\tau\ a^{\downarrow}[hom_o\ a,\ F_o\ a]) \cdot hom_a\ x\ a$
using *assms x C.DN-def* **by** *fastforce*
also have ... =
$eval\ \mathcal{I}\ (F_o\ a) \cdot \mathrm{r}^{-1}[exp\ \mathcal{I}\ (F_o\ a)] \cdot C.Exp\ (Id\ x)\ (F_o\ a) \cdot$
$Exp^{\leftarrow}\ (\tau\ x^{\downarrow}[hom_o\ x,\ F_o\ x])\ (F_o\ a) \cdot F_a\ x\ a$
using *assms*(*2*) *left-square* τ*.enriched-natural-transformation-axioms*
by *fastforce*
also have ... =
$eval\ \mathcal{I}\ (F_o\ a) \cdot \mathrm{r}^{-1}[exp\ \mathcal{I}\ (F_o\ a)] \cdot C.Exp\ (Id\ x)\ (F_o\ a) \cdot$
$Exp^{\leftarrow}\ (Uncurry[hom_o\ x,\ F_o\ x]\ (\tau\ x) \cdot \mathrm{l}^{-1}[hom_o\ x])\ (F_o\ a) \cdot F_a\ x\ a$
using *C.DN-def* **by** *fastforce*
finally show *?thesis* **by** *blast*
qed
also have ... =
$eval\ \mathcal{I}\ (F_o\ a) \cdot \mathrm{r}^{-1}[exp\ \mathcal{I}\ (F_o\ a)] \cdot$
$(C.Exp\ (Id\ x)\ (F_o\ a) \cdot$
$Exp^{\leftarrow}\ (Uncurry[hom_o\ x,\ F_o\ x]\ (\tau\ x) \cdot \mathrm{l}^{-1}[hom_o\ x])\ (F_o\ a)) \cdot$
$F_a\ x\ a$
using *comp-assoc* **by** *simp*
also have ... =
$eval\ \mathcal{I}\ (F_o\ a) \cdot \mathrm{r}^{-1}[exp\ \mathcal{I}\ (F_o\ a)] \cdot$
$(Exp^{\leftarrow}\ (Id\ x)\ (F_o\ a) \cdot$
$Exp^{\leftarrow}\ (Uncurry[hom_o\ x,\ F_o\ x]\ (\tau\ x) \cdot \mathrm{l}^{-1}[hom_o\ x])\ (F_o\ a)) \cdot$
$F_a\ x\ a$
using *assms x F.preserves-Obj C.Exp-def C.cov-Exp-ide*
comp-cod-arr [*of* $Exp^{\leftarrow}\ (Id\ x)\ (dom\ (F_o\ a))$]
by *auto*
also have ... =
$eval\ \mathcal{I}\ (F_o\ a) \cdot \mathrm{r}^{-1}[exp\ \mathcal{I}\ (F_o\ a)] \cdot$
$Exp^{\leftarrow}\ ((Uncurry[hom_o\ x,\ F_o\ x]\ (\tau\ x) \cdot \mathrm{l}^{-1}[hom_o\ x]) \cdot Id\ x)\ (F_o\ a) \cdot$
$F_a\ x\ a$
proof −
have $seq\ (Uncurry[hom_o\ x,\ F_o\ x]\ (\tau\ x) \cdot \mathrm{l}^{-1}[hom_o\ x])\ (Id\ x)$
using *assms x F.preserves-Obj Id-in-hom* τ*.component-in-hom*
apply (*intro seqI*)
apply *auto*[*1*]
by *force+*
thus *?thesis*
using *assms x F.preserves-Obj C.cnt-Exp-comp* **by** *simp*
qed
also have ... = $eval\ \mathcal{I}\ (F_o\ a) \cdot \mathrm{r}^{-1}[exp\ \mathcal{I}\ (F_o\ a)] \cdot$
$Exp^{\leftarrow}\ (generating\text{-}elem\ \tau)\ (F_o\ a) \cdot F_a\ x\ a$
using *x C.DN-def comp-assoc* τ*.component-in-hom* **by** *fastforce*
also have *1*: ... =

(*generated-transformation* (*generating-elem* τ) *a*) $\downarrow$[*hom*$_o$ *a*, *F*$_o$ *a*]
using *assms x F.preserves-Obj C.UP-DN(4) τ.component-in-hom calculation ide-Hom*
by (*metis* (*no-types*, *lifting*) *mem-Collect-eq*)
finally have $*$: (τ *a*) $\downarrow$[*hom*$_o$ *a*, *F*$_o$ *a*] =
(*generated-transformation* (*generating-elem* τ) *a*)
$\downarrow$[*hom*$_o$ *a*, *F*$_o$ *a*]
by *blast*
have τ *a* = ((τ *a*) $\downarrow$[*hom*$_o$ *a*, *F*$_o$ *a*])$^\uparrow$
using *assms x τ.component-in-hom ide-Hom F.preserves-Obj* **by** *auto*
also have ... = ((*generated-transformation* (*generating-elem* τ) *a*)
$\downarrow$[*hom*$_o$ *a*, *F*$_o$ *a*])$^\uparrow$
using $*$ **by** *argo*
also have ... = *generated-transformation* (*generating-elem* τ) *a*
using *assms x 1 ide-Hom* **by** *presburger*
finally show τ *a* = *generated-transformation* (*generating-elem* τ) *a* **by** *blast*
qed

lemma *element-of-generated-transformation*:
assumes *e* $\in$ *hom* $\mathcal{I}$ (*F*$_o$ *x*)
shows *generating-elem* (*generated-transformation e*) = *e*
proof –
have *generating-elem* (*generated-transformation e*) =
Uncurry[*hom*$_o$ *x*, *F*$_o$ *x*]
((*eval* $\mathcal{I}$ (*F*$_o$ *x*) · r^{-1}[*exp* $\mathcal{I}$ (*F*$_o$ *x*)] ·
Curry[*exp* (*F*$_o$ *x*) (*F*$_o$ *x*), $\mathcal{I}$, *F*$_o$ *x*]
(*eval* (*F*$_o$ *x*) (*F*$_o$ *x*) · (*exp* (*F*$_o$ *x*) (*F*$_o$ *x*) $\otimes$ *e*)) · *F*$_a$ *x x*)$^\uparrow$) ·
l^{-1}[*hom*$_o$ *x*] · *Id x*
proof –
have *arr* ((*eval* $\mathcal{I}$ (*F*$_o$ *x*) ·
r^{-1}[*exp* $\mathcal{I}$ (*F*$_o$ *x*)] ·
Curry[*exp* (*F*$_o$ *x*) (*F*$_o$ *x*), $\mathcal{I}$, *F*$_o$ *x*]
(*eval* (*F*$_o$ *x*) (*F*$_o$ *x*) · (*exp* (*F*$_o$ *x*) (*F*$_o$ *x*) $\otimes$ *e*)) · *F*$_a$ *x x*)$^\uparrow$)
using *assms x F.preserves-Hom F.preserves-Obj*
apply (*intro C.UP-simps seqI*)
apply *auto*[*1*]
by *fastforce+*
thus *?thesis*
using *assms x C.DN-def comp-assoc* **by** *auto*
qed
also have ... =
Uncurry[*hom*$_o$ *x*, *F*$_o$ *x*]
((*eval* $\mathcal{I}$ (*F*$_o$ *x*) ·
(r^{-1}[*exp* $\mathcal{I}$ (*F*$_o$ *x*)] ·
Curry[*exp* (*F*$_o$ *x*) (*F*$_o$ *x*), $\mathcal{I}$, *F*$_o$ *x*]
(*eval* (*F*$_o$ *x*) (*F*$_o$ *x*) · (*exp* (*F*$_o$ *x*) (*F*$_o$ *x*) $\otimes$ *e*))) · *F*$_a$ *x x*)$^\uparrow$) ·
l^{-1}[*hom*$_o$ *x*] · *Id x*
using *comp-assoc* **by** *simp*
also have ... =

$Uncurry[hom_o\ x,\ F_o\ x]$
$((eval\ \mathcal{I}\ (F_o\ x)\ \cdot$
$((Curry[exp\ (F_o\ x)\ (F_o\ x),\ \mathcal{I},\ F_o\ x]$
$(eval\ (F_o\ x)\ (F_o\ x) \cdot (exp\ (F_o\ x)\ (F_o\ x) \otimes e)) \otimes \mathcal{I}) \cdot$
$\mathrm{r}^{-1}[exp\ (F_o\ x)\ (F_o\ x)]) \cdot$
$F_a\ x\ x)^{\uparrow}) \cdot$
$\mathrm{l}^{-1}[hom_o\ x] \cdot Id\ x$
proof −
have «$Curry[exp\ (F_o\ x)\ (F_o\ x),\ \mathcal{I},\ F_o\ x]$
$(eval\ (F_o\ x)\ (F_o\ x) \cdot (exp\ (F_o\ x)\ (F_o\ x) \otimes e))$
$: exp\ (F_o\ x)\ (F_o\ x) \rightarrow exp\ \mathcal{I}\ (F_o\ x)$»
using *assms x F.preserves-Obj C.ide-exp*
by (*intro C.Curry-in-hom*) *auto*
thus *?thesis*
using *assms*
runit′-naturality
[*of* $Curry[exp\ (F_o\ x)\ (F_o\ x),\ \mathcal{I},\ F_o\ x]$
$(eval\ (F_o\ x)\ (F_o\ x) \cdot (exp\ (F_o\ x)\ (F_o\ x) \otimes e))]$
by *force*
qed
also have ... =
$Uncurry[hom_o\ x,\ F_o\ x]$
$((Uncurry[\mathcal{I},\ F_o\ x]$
$(Curry[exp\ (F_o\ x)\ (F_o\ x),\ \mathcal{I},\ F_o\ x]$
$(eval\ (F_o\ x)\ (F_o\ x) \cdot (exp\ (F_o\ x)\ (F_o\ x) \otimes e))) \cdot$
$\mathrm{r}^{-1}[exp\ (F_o\ x)\ (F_o\ x)] \cdot F_a\ x\ x)^{\uparrow}) \cdot$
$\mathrm{l}^{-1}[hom_o\ x] \cdot Id\ x$
using *comp-assoc* **by** *simp*
also have ... =
$Uncurry[hom_o\ x,\ F_o\ x]$
$(((eval\ (F_o\ x)\ (F_o\ x) \cdot (exp\ (F_o\ x)\ (F_o\ x) \otimes e)) \cdot$
$\mathrm{r}^{-1}[exp\ (F_o\ x)\ (F_o\ x)] \cdot F_a\ x\ x)^{\uparrow}) \cdot$
$\mathrm{l}^{-1}[hom_o\ x] \cdot Id\ x$
using *assms x F.preserves-Obj C.Uncurry-Curry* **by** *auto*
also have ... =
$Uncurry[hom_o\ x,\ F_o\ x]$
$(((eval\ (F_o\ x)\ (F_o\ x) \cdot (exp\ (F_o\ x)\ (F_o\ x) \otimes e)) \cdot$
$(F_a\ x\ x \otimes \mathcal{I}) \cdot \mathrm{r}^{-1}[hom_o\ x])^{\uparrow}) \cdot$
$\mathrm{l}^{-1}[hom_o\ x] \cdot Id\ x$
using *assms x runit′-naturality F.preserves-Hom* [*of x x*] **by** *fastforce*
also have ... =
$Uncurry[hom_o\ x,\ F_o\ x]$
$(((eval\ (F_o\ x)\ (F_o\ x) \cdot (exp\ (F_o\ x)\ (F_o\ x) \otimes e) \cdot (F_a\ x\ x \otimes \mathcal{I})) \cdot$
$\mathrm{r}^{-1}[hom_o\ x])^{\uparrow}) \cdot$
$\mathrm{l}^{-1}[hom_o\ x] \cdot Id\ x$
using *comp-assoc* **by** *simp*
also have ... =
$Uncurry[hom_o\ x,\ F_o\ x]$
$(((eval\ (F_o\ x)\ (F_o\ x) \cdot (F_a\ x\ x \otimes e)) \cdot \mathrm{r}^{-1}[hom_o\ x])^{\uparrow}) \cdot$

$l^{-1}[hom_o\ x] \cdot Id\ x$
using *assms x F.preserves-Hom [of x x] comp-arr-dom [of e* $\mathcal{I}$*] comp-cod-arr interchange*
by *fastforce*
also have ... =
$Uncurry[hom_o\ x,\ F_o\ x]$
$(Curry[\mathcal{I},\ hom_o\ x,\ F_o\ x]$
$(((eval\ (F_o\ x)\ (F_o\ x) \cdot (F_a\ x\ x \otimes e)) \cdot r^{-1}[hom_o\ x]) \cdot l[hom_o\ x])) \cdot$
$l^{-1}[hom_o\ x] \cdot Id\ x$
proof −
have *seq* $(eval\ (F_o\ x)\ (F_o\ x) \cdot (F_a\ x\ x \otimes e))\ r^{-1}[Hom\ x\ x]$
using *assms x F.preserves-Obj F.preserves-Hom* **by** *blast*
thus *?thesis*
using *assms x C.UP-def F.preserves-Obj* **by** *auto*
qed
also have ... =
$(((eval\ (F_o\ x)\ (F_o\ x) \cdot (F_a\ x\ x \otimes e)) \cdot r^{-1}[Hom\ x\ x]) \cdot l[Hom\ x\ x]) \cdot$
$l^{-1}[Hom\ x\ x] \cdot Id\ x$
using *assms x C.Uncurry-Curry F.preserves-Obj F.preserves-Hom* **by** *force*
also have ... =
$eval\ (F_o\ x)\ (F_o\ x) \cdot (F_a\ x\ x \otimes e) \cdot r^{-1}[Hom\ x\ x] \cdot$
$(l[Hom\ x\ x] \cdot l^{-1}[Hom\ x\ x]) \cdot Id\ x$
using *comp-assoc* **by** *simp*
also have ... = $eval\ (F_o\ x)\ (F_o\ x) \cdot (F_a\ x\ x \otimes e) \cdot r^{-1}[Hom\ x\ x] \cdot Id\ x$
using *assms x ide-Hom Id-in-hom comp-lunit-lunit'(1) comp-cod-arr*
by *fastforce*
also have ... = $eval\ (F_o\ x)\ (F_o\ x) \cdot (F_a\ x\ x \otimes e) \cdot (Id\ x \otimes \mathcal{I}) \cdot r^{-1}[\mathcal{I}]$
using *x Id-in-hom runit'-naturality* **by** *fastforce*
also have ... = $eval\ (F_o\ x)\ (F_o\ x) \cdot ((F_a\ x\ x \otimes e) \cdot (Id\ x \otimes \mathcal{I})) \cdot r^{-1}[\mathcal{I}]$
using *comp-assoc* **by** *simp*
also have ... = $eval\ (F_o\ x)\ (F_o\ x) \cdot (F_a\ x\ x \cdot Id\ x \otimes e) \cdot r^{-1}[\mathcal{I}]$
using *assms x interchange [of* F_a *x x Id x e* $\mathcal{I}$*] F.preserves-Hom comp-arr-dom Id-in-hom*
by *fastforce*
also have ... = $eval\ (F_o\ x)\ (F_o\ x) \cdot (C.Id\ (F_o\ x) \otimes e) \cdot r^{-1}[\mathcal{I}]$
using *x F.preserves-Id* **by** *auto*
also have ... =
$eval\ (F_o\ x)\ (F_o\ x) \cdot ((C.Id\ (F_o\ x) \otimes F_o\ x) \cdot (\mathcal{I} \otimes e)) \cdot r^{-1}[\mathcal{I}]$
using *assms x interchange C.Id-in-hom F.preserves-Obj comp-arr-dom comp-cod-arr*
by *(metis in-homE mem-Collect-eq)*
also have ... = $Uncurry[F_o\ x,\ F_o\ x]\ (C.Id\ (F_o\ x)) \cdot (\mathcal{I} \otimes e) \cdot r^{-1}[\mathcal{I}]$
using *comp-assoc* **by** *simp*
also have ... = $l[F_o\ x] \cdot (\mathcal{I} \otimes e) \cdot r^{-1}[\mathcal{I}]$
using *x F.preserves-Obj C.Id-def C.Uncurry-Curry* **by** *fastforce*
also have ... = $l[F_o\ x] \cdot (\mathcal{I} \otimes e) \cdot l^{-1}[\mathcal{I}]$
using *unitor-coincidence* **by** *simp*
also have ... = $l[F_o\ x] \cdot l^{-1}[F_o\ x] \cdot e$
using *assms lunit'-naturality* **by** *fastforce*

```
    also have ... = (l[F_o x] · l⁻¹[F_o x]) · e
      using comp-assoc by simp
    also have ... = e
      using assms x comp-lunit-lunit' F.preserves-Obj comp-cod-arr by auto
    finally show generating-elem (generated-transformation e) = e
      by blast
  qed
```

We can now state and prove the (weak) covariant Yoneda lemma (Kelly, Section 1.9) for enriched categories.

```
  theorem covariant-yoneda:
  shows bij-betw generated-transformation
          (hom I (F_o x))
          (Collect (enriched-natural-transformation C T α ι
                      Obj Hom Id Comp (Collect ide) exp C.Id C.Comp
                      hom_o hom_a F_o F_a))
  proof (intro bij-betwI)
    show generated-transformation ∈
           hom I (F_o x) → Collect
                             (enriched-natural-transformation C T α ι
                                Obj Hom Id Comp (Collect ide) exp C.Id C.Comp
                                hom_o hom_a F_o F_a)
      using enriched-natural-transformation-generated-transformation by blast
    show generating-elem ∈
           Collect (enriched-natural-transformation C T α ι
                      Obj Hom Id Comp (Collect ide) exp C.Id C.Comp
                      hom_o hom_a F_o F_a)
             → hom I (F_o x)
      using generating-elem-in-hom by blast
    show ⋀e. e ∈ hom I (F_o x) ⟹
               generating-elem (generated-transformation e) = e
      using element-of-generated-transformation by blast
    show ⋀τ. τ ∈ Collect (enriched-natural-transformation C T α ι
                            Obj Hom Id Comp (Collect ide) exp C.Id C.Comp
                            hom_o hom_a F_o F_a)
              ⟹ generated-transformation (generating-elem τ) = τ
    proof -
      fix τ
      assume τ: τ ∈ Collect (enriched-natural-transformation C T α ι
                               Obj Hom Id Comp (Collect ide) exp C.Id C.Comp
                               hom_o hom_a F_o F_a)
      interpret τ: enriched-natural-transformation C T α ι
                     Obj Hom Id Comp ‹Collect ide› exp C.Id C.Comp
                     hom_o hom_a F_o F_a τ
        using τ by blast
      show generated-transformation (generating-elem τ) = τ
      proof
        fix a
        show generated-transformation (generating-elem τ) a = τ a
```

using τ *transformation-generated-by-element* τ*.extensionality*
F.extensionality C.UP-def not-arr-null null-is-zero(*2*)
by (*cases a* ∈ *Obj*) *auto*
qed
qed
qed

end

2.5.3 Contravariant Case

The (weak) contravariant Yoneda lemma is obtained by just replacing the enriched category by its opposite in the covariant version.

locale *contravariant-yoneda-lemma* =
opposite-enriched-category $C\ T\ \alpha\ \iota\ \sigma$ *Obj Hom Id Comp* +
covariant-yoneda-lemma $C\ T\ \alpha\ \iota\ \sigma$ *exp eval Curry Obj* Hom_{op} *Id* $Comp_{op}$ *y* F_o F_a
for C :: $'a \Rightarrow 'a \Rightarrow 'a$ (**infixr** ‹·› *55*)
and T :: $'a \times 'a \Rightarrow 'a$
and α :: $'a \times 'a \times 'a \Rightarrow 'a$
and ι :: $'a$
and σ :: $'a \times 'a \Rightarrow 'a$
and *exp* :: $'a \Rightarrow 'a \Rightarrow 'a$
and *eval* :: $'a \Rightarrow 'a \Rightarrow 'a$
and *Curry* :: $'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a \Rightarrow 'a$
and *Obj* :: $'b$ *set*
and *Hom* :: $'b \Rightarrow 'b \Rightarrow 'a$
and *Id* :: $'b \Rightarrow 'a$
and *Comp* :: $'b \Rightarrow 'b \Rightarrow 'b \Rightarrow 'a$
and *y* :: $'b$
and F_o :: $'b \Rightarrow 'a$
and F_a :: $'b \Rightarrow 'b \Rightarrow 'a$
begin

corollary *contravariant-yoneda*:
shows *bij-betw generated-transformation*
(*hom* $\mathcal{I}$ (F_o *y*))
(*Collect*
(*enriched-natural-transformation*
$C\ T\ \alpha\ \iota$ *Obj* Hom_{op} *Id* $Comp_{op}$ (*Collect ide*) *exp C.Id C.Comp*
hom_o hom_a F_o F_a))
using *covariant-yoneda* **by** *blast*

end

end

Bibliography

[1] G. M. Kelly. Basic concepts of enriched category theory. *Reprints in Theory and Applications of Categories*, 10, 2005. http://www.tac.mta.ca/tac/reprints/articles/10/tr10.pdf.

[2] nLab. internal hom. *nLab (various contributors)*, 2009 – 2024. https://ncatlab.org/nlab/show/internal+hom, [Online; accessed 22-May-2024].

[3] E. W. Stark. Monoidal categories. *Archive of Formal Proofs*, May 2017. https://isa-afp.org/entries/MonoidalCategory.html, Formal proof development.

[4] E. W. Stark. Residuated transition systems II: Categorical properties. *Archive of Formal Proofs*, June 2024. (submitted for publication).