USER GUIDE

SCOTT MORRISON AND DAVID SPIVAK

1. Scala

1.1. Class hierarchies. This section contains an overview of the important types
in the Scala library. You should read this section in conjunction with the Scaladocs.

The three most important types are

- net.metaphor.api.**Ontology**
    All database schemas have type **Ontology**.
- net.metaphor.api.**Translation**
    A **Translation** is a functor between two **Ontology**s
- net.metaphor.api.**Ontology#Dataset**
    A **Dataset** is a functor from an **Ontology** to **Set**. (Recall in Scala the
    # denotes an inner class — thus every **Dataset** is attached to a particular
    Ontology instance.)

Ontology is a subtype of a long sequence of more general classes of categories,
illustrated in Figure 1.1.

```
trait Category {
    type O
    type M

    def identity(o: O): M
    def source(m: M): O
    def target(m: M): O
    def compose(m1: M, m2: M): M
    ...
}
```

```
trait SmallCategory {
    trait FunctorToSet { ... }
    ...
}
```

```
trait LocallyFinitelyGeneratedCategory {
    type G
    override type M = PathEquivalenceClass
    def pathEquality(p1: Path, p2: Path): Boolean

    override def compose(m1: M, m2: M) = ...

    def objectsAtLevel(k: Int): List[O]
    val minimumLevel: Int
    def generators(s: O, t: O): List[G]
    ...
}
```

```
trait FinitelyGeneratedCategory {
    val maximumLevel: Int
    ...
}
```

```
trait FinitelyPresentedCategory {
    def relations(s: O, t: O): List[(Path, Path)]
    ...
}
```
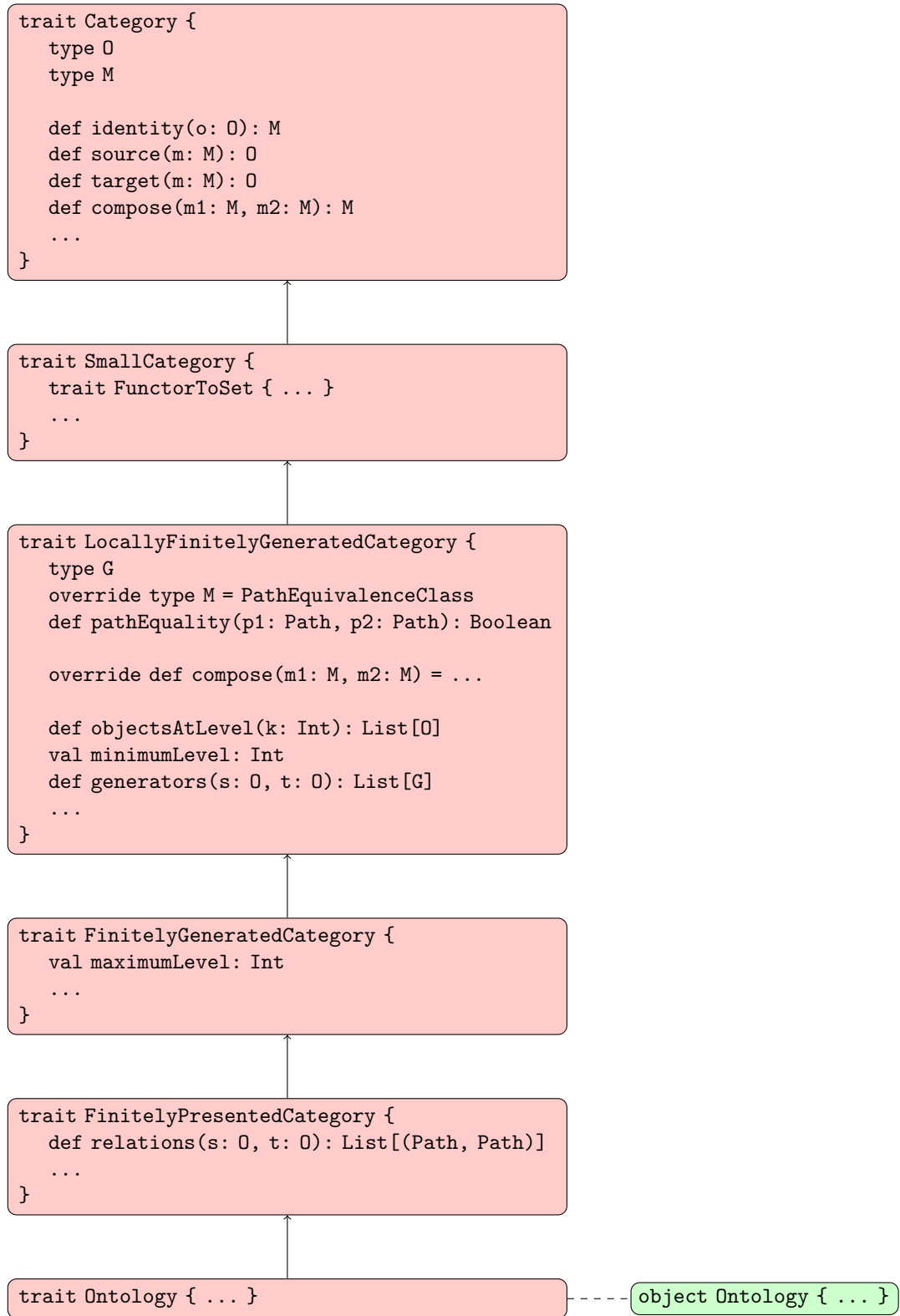
```
trait Ontology { ... }
```   ```
object Ontology { ... }
```

FIGURE 1. The Ontology type hierarchy.