Semantics of Programming Languages
Exercise Sheet 12

In this exercise, you will prove some properties of Galois connections—these provide a general framework for the abstract interpretation concretization functions, such as the for parity analysis from the lectures.

We start by defining a slightly enriched version of the parity domain from the lectures.

```plaintext
datatype parity = Even | Odd | Either | Emp
```

Recall that `Even`, `Odd` and `Either` are aimed at representing information about sets of integers: consisting of only even numbers, only odd numbers, and either odd or even numbers, respectively. We have added a fourth element, `Emp`, covering the case of a set both consisting of only even numbers and consisting of only odd number—the only such set is the empty set.

Moreover, recall that we model the notion of containing more information as an order relation, here denoted $\leq$. Thus, e.g., we need to set $\text{Even} \leq \text{Either}$.

Your first task is to define $\leq$ on the elements of `parity` and then register `parity` as a partial order.

```plaintext
instantiation parity :: order
≤ is more than a partial order: it is lattice, even a bounded one, and even a complete one. Climb the type-class hierarchy with `parity` by defining the necessary operators and proving the necessary facts.
```

```plaintext
instantiation parity :: lattice
instantiation parity :: bounded_lattice
instantiation parity :: complete_lattice
```

Extend the concretization function from the lectures by mapping `Emp` to the empty set:

```plaintext
fun γ_parity :: “parity ⇒ int set” where
```

Think of how one came up the domain `parity`: the four values of `parity` represent degrees of knowledge about the parity of the integers in given sets. This intuition can be modelled by a so-called abstraction function:

```plaintext
definition α_parity :: “int set ⇒ parity” where
```
Sometimes the desired concretization function is definable from an abstraction function—this is the case here:

**lemma** \( \gamma_{\text{parity}} \alpha_{\text{parity}} \): \( \gamma_{\text{parity}} a = \bigcup \{ S . \alpha_{\text{parity}} S \leq a \} \)

Intuitively, we read \( \alpha_{\text{parity}} S \leq x \) as \( x \) approximates \( S \). (Indeed, according to the intuitive reading of \( \leq \), \( \alpha_{\text{parity}} S \) contains less information than \( x \).) Then \( \gamma_{\text{parity}} x \) can be taken to be the largest set approximated by \( x \).

Dually, \( \alpha_{\text{parity}} \) can be obtained from \( \gamma_{\text{parity}} \), defining \( \alpha_{\text{parity}} S \) to be the best approximation of \( S \) w.r.t. \( \gamma_{\text{parity}} \):

**lemma** \( \alpha_{\text{parity}} \gamma_{\text{parity}} \): \( \alpha_{\text{parity}} S = \text{Inf} \{ a . S \subseteq \gamma_{\text{parity}} a \} \)

The above properties can be obtained more abstractly, using the concept of Galois connection:

**definition** \( \text{galois} :: \)

\[
\left( (\alpha :: \text{complete lattice} \Rightarrow c :: \text{complete lattice}) \Rightarrow (c \Rightarrow 'a) \Rightarrow \text{bool} \right)
\]

**where**

\( \text{galois} \alpha \gamma \equiv \forall c a. \alpha c \leq a \iff c \leq \gamma a \)

Galois connections postulate the ideal relationship between a concretization and an abstraction: \( x \) approximates the abstraction of \( s \) iff the concretization of \( x \) approximates \( s \).

Show that \( \alpha_{\text{parity}} \) and \( \gamma_{\text{parity}} \) form a Galois connection:

**lemma** \( \text{galois} \alpha_{\text{parity}} \gamma_{\text{parity}} \)

Prove the following consequences of the Galois connection property, including monotonicity of its components and the very facts we have proved about \( \alpha_{\text{parity}} \) and \( \gamma_{\text{parity}} \):

**lemma** \( \gamma_{\alpha} \text{infl} \): assumes \( \text{galois} \alpha \gamma \) shows \( c \leq \gamma(\alpha c) \)

**lemma** \( \alpha \gamma \text{defl} \): assumes \( \text{galois} \alpha \gamma \) shows \( \alpha(\gamma a) \leq a \)

**lemma** \( \gamma_{\alpha} \text{mono} \): assumes \( \text{galois} \alpha \gamma \) shows \( \text{mono} \gamma \)

**lemma** \( \alpha_{\gamma} \text{mono} \): assumes \( \text{galois} \alpha \gamma \) shows \( \text{mono} \alpha \)

**lemma** \( \gamma_{\alpha} \text{x} \): assumes \( \text{galois} \alpha \gamma \) shows \( \gamma a = \text{Sup} \{ c . \alpha c \leq a \} \)

**lemma** \( \alpha_{\gamma} \text{x} \): assumes \( \text{galois} \alpha \gamma \) shows \( \alpha c = \text{Inf} \{ a . c \leq \gamma a \} \)