

Domácí úkol 4: Relace

V tomto domácím úkolu si vyzkoušíte vytvořit o něco delší kód v Haskellu, odměnou vám za to mohou být až dva body do hodnocení. Za úkol máte naprogramovat několik funkcí pracujících s binárními relacemi. Pokud si nepamätujete (nebo jste se dosud nedozvěděli), co jsou to relace, doporučujeme se dozdělat třeba na Wikipedii. Potřebné definice však budou zmíněny i u popisu jednotlivých funkcí.

Pro jednoduchost pracujeme pouze s relacemi nad jedinou množinou, jíž jsou písmena malé anglické abecedy. Jejich seznam máte k dispozici v konstantě `letters :: [Char]`.

Kostru domácí úlohy si můžete stáhnout ze studijních materiálů.

Reprezentace relací

Naše relace budou reprezentovány seznamy dvojic znaků. Přitom platí, že:

- Prvky x a y jsou v relaci R právě tehdy, když seznam R obsahuje dvojici (x, y) . Zapisujeme $(x, y) \in R$.
- Na pořadí dvojic v seznamu nezáleží.
- Na pořadí prvků ve dvojici samozřejmě záleží.
- **Pozor:** stejná dvojice se v seznamu nesmí objevit více než jednou.

Pro zpřehlednění kódu zavádíme typový alias `Relation`:

```
type Relation = [(Char, Char)]
```

Takto zadefinovaný typ `Relation` je zcela identický s typem `[(Char, Char)]`. Něco podobného už znáte: typ `String` je také pouhým aliasem na `[Char]`.

Můžete předpokládat, že funkce, které berou jako parametr `Relation`, budou mít na vstupu vždy přípustné relace (tedy bez duplicitních prvků).

Funkce k implementaci

Následuje popis funkcí, které máte implementovat. V příkladech se držíme zavedeného formátu, v němž řádky začínající znakem „>“ označují vstup do interpretru a ostatní řádky jsou jeho výstupem.

• `idRel :: Relation`

Relace identity. Každý znak je v relaci právě a pouze sám se sebou.

Příklad:

```
> take 3 idRel
[('a', 'a'), ('b', 'b'), ('c', 'c')]
```

• `union :: Relation -> Relation -> Relation`

Vrací sjednocení dvou relací. *Sjednocením* relací R a S je relace T taková, že pro každé dva prvky x, y platí $(x, y) \in T \iff (x, y) \in R \vee (x, y) \in S$.

Příklad:

```
> [( 'a', 'b' )] `union` [( 'b', 'a' )]
[('a', 'b'), ('b', 'a')]
> [( 'a', 'b' ), ('c', 'd' )] `union` [( 'b', 'a' ), ('c', 'd' )]
[('a', 'b'), ('c', 'd'), ('b', 'a')]
> [( 'a', 'c' )] `union` []
[('a', 'c')]
```

• **intersection :: Relation -> Relation -> Relation**

Vrací průnik dvou relací. *Průnikem* relací R a S je relace T taková, že pro každé dva prvky x, y platí $(x, y) \in T \iff (x, y) \in R \wedge (x, y) \in S$.

Příklad:

```
> [('a', 'b')] `intersection` [('b', 'a')]
[]
> [('a', 'b'), ('c', 'd')] `intersection` [('b', 'a'), ('c', 'd')]
[('c', 'd')]
```

• **equal :: Relation -> Relation -> Bool**

Rozhodne, zda jsou dvě relace totožné; to jest, vrací True, pokud zadané relace obsahují právě stejné dvojice prvků. Přitom nezáleží na pořadí těchto dvojic.

Příklad:

```
> [('a', 'b'), ('c', 'd')] `equal` [('a', 'b'), ('c', 'd')]
True
> [('c', 'd'), ('a', 'b')] `equal` [('a', 'b'), ('c', 'd')]
True
> [('a', 'b')] `equal` [('b', 'a')]
False
> [('a', 'b'), ('c', 'd')] `equal` [('b', 'a'), ('c', 'd')]
False
> equal [] []
True
```

• **composition :: Relation -> Relation -> Relation**

Spočítá složení dvou relací. *Složením* relací $R \circ S$ je relace T taková, že pro každé dva prvky x, y platí $(x, y) \in T \iff \exists z : (x, z) \in S \wedge (z, y) \in R$.

Příklad:

```
> [('a', 'b'), ('c', 'd')] `composition` [('b', 'a'), ('c', 'a')]
[('b', 'b'), ('c', 'b')]
> [('c', 'd'), ('a', 'b')] `composition` [('a', 'b'), ('c', 'd')]
[]
> [('a', 'a'), ('a', 'b'), ('a', 'd')] `composition` [('b', 'a')]
[('b', 'a'), ('b', 'b'), ('b', 'd')]
> [('a', 'b'), ('c', 'b')] `composition` [('b', 'a'), ('b', 'c')]
[('b', 'b')]
```

• **power :: Relation -> Int -> Relation**

Spočítá mocninu relace. Exponent relace bude vždy nezáporný; neplatné vstupy nemusíte ošetřovat. N -tá *mocnina* relace je rekurzivně definována takto:

- Nultá mocnina jakékoli relace je relace identity.
- Pro $n > 0$ platí $R^n = R \circ R^{n-1}$.

Příklad:

```
> [('a', 'b'), ('b', 'c'), ('c', 'd')] `power` 1
[('a', 'b'), ('b', 'c'), ('c', 'd')]
> [('a', 'b'), ('b', 'c'), ('c', 'd')] `power` 2
[('a', 'c'), ('b', 'd')]
> [('a', 'b'), ('b', 'c'), ('c', 'd')] `power` 3
```

```

[('a', 'd')]
> [('a', 'b'), ('b', 'c'), ('c', 'd')] `power` 4
[]
> idRel `equal` ([('a', 'b'), ('b', 'c'), ('c', 'd')] `power` 0)
True

```

• **inverse :: Relation -> Relation**

Pro zadanou relaci vrátí její inverzi. *Inverzí* relace R je relace R^{-1} , kde pro každé dva prvky x, y platí:
 $(x, y) \in R^{-1} \iff (y, x) \in R$.

Příklad:

```

> inverse [('a', 'b'), ('b', 'c'), ('c', 'd')]
[('b', 'a'), ('c', 'b'), ('d', 'c')]
> inverse [('a', 'a'), ('a', 'b'), ('b', 'a')]
[('a', 'a'), ('b', 'a'), ('a', 'b')]
> inverse idRel `equal` idRel
True

```

• **isReflexive :: Relation -> Bool**

Zjistí, zda je daná relace reflexivní. Relace R je reflexivní tehdy, pokud pro každý prvek x z nosné množiny platí $(x, x) \in R$. Připomenutí: nosnou množinou jsou všechna malá písmena anglické abecedy (**letters**).

Příklad:

```

> isReflexive [('a', 'b'), ('b', 'c'), ('c', 'd')]
False
> isReflexive [('a', 'a'), ('b', 'b'), ('c', 'c')]
False
> isReflexive idRel
True
> isReflexive (idRel `union` [('a', 'b'), ('c', 'a')])
True
> isReflexive []
False

```

• **isSymmetric :: Relation -> Bool**

Zjistí, zda je daná relace symetrická. Relace R je symetrická tehdy, pokud pro každé dva prvky x, y platí
 $(x, y) \in R \implies (y, x) \in R$.

Příklad:

```

> isSymmetric [('a', 'b'), ('b', 'c'), ('c', 'a')]
False
> isSymmetric [('a', 'b'), ('b', 'a'), ('c', 'c')]
True
> isSymmetric [('a', 'a'), ('b', 'b'), ('c', 'c')]
True
> isSymmetric idRel
True
> isSymmetric []
True

```

• `isTransitive :: Relation -> Bool`

Zjistí, zda je daná relace tranzitivní. Relace R je tranzitivní tehdy, pokud pro každé tři prvky x, y, z platí $(x, y) \in R \wedge (y, z) \in R \implies (x, z) \in R$.

Příklad:

```
> isTransitive [('a', 'b'), ('b', 'c'), ('c', 'd')]
False
> isTransitive [('a', 'b'), ('b', 'c'), ('a', 'c')]
True
> isTransitive [('a', 'b'), ('c', 'd'), ('e', 'f')]
True
> isTransitive [('a', 'b'), ('b', 'a'), ('b', 'd')]
False
> isTransitive idRel
True
> isTransitive []
True
```

• `isAntisymmetric :: Relation -> Bool`

Zjistí, zda je daná relace antisymetrická. Relace R je antisymetrická tehdy, pokud pro každé dva prvky x, y platí $(x, y) \in R \wedge (y, x) \in R \implies x = y$.

Příklad:

```
> isAntisymmetric [('a', 'b'), ('b', 'c'), ('c', 'd')]
True
> isAntisymmetric [('a', 'b'), ('b', 'a'), ('c', 'd')]
False
> isAntisymmetric [('a', 'b'), ('c', 'd'), ('f', 'e')]
True
> isAntisymmetric [('a', 'a'), ('b', 'a'), ('c', 'd'), ('a', 'b')]
False
> isAntisymmetric idRel
True
> isAntisymmetric []
True
```

Poznámky a tipy

- Podívejte se na funkce `nub` a `sort` z modulu `Data.List`, budou se vám hodit.
- Vhod mohou přijít také funkce `and`, `or`, `all` a `any` z `Prelude`.
- Kromě zmiňovaných dvou funkcí z modulu `Data.List` nesmíte importovat žádné další funkce a moduly.
- Práci vám ohromně usnadní intenzionální zápis seznamů.
- Směle využívejte svých již definovaných funkcí.
- Přebíráte-li kód odjinud, nezapomeňte uvést zdroj. V opačném případě bude na vaši práci pohlíženo jako na plagiát.

Odevzdání a bodování

Domácí úkol se odevzdává přes odpovědník v ISu. Tento odpovědník obsahuje jediné pole, tam vložte svou implementaci požadovaných funkcí, včetně importu a definic z kostry řešení. Své řešení komentujte v kódu, tato úloha totiž podléhá také ruční kontrole cvičícími, při níž se přihlíží i k popisu řešení.

Máte **dvě možnosti odevzdání** (a samozřejmě kontrolu syntaxe před odevzdáním). Dokud nekliknete na „odevdat“, můžete odpovědník zavřít a znovu otevřít bez penalizace. Nezapomeňte na průběžné ukládání. Po každém odevzdání uvidíte výsledky automatických testů, věnujte však čas dostatečnému vlastnímu testování.

Pokud vaše řešení projde všemi automatickými testy, obdržíte jeden bod. V případě, že se vám podařilo implementovat jen část funkcí, sice žádný bod automaticky nedostanete, ale cvičící vám podle míry vyřešenosti při ruční kontrole úlohy udělí část bodu. Za kvalitu kódu a jeho popis vám cvičící mohou udělit další jeden bod nebo jeho část.

V součtu tedy můžete za úlohu získat až 2 body.