Automatische Bewertung von Haskell-Programmieraufgaben

Marcellus Siegburg, Janis Voigtländer, Oliver Westphal ▪ 08.10.2019

Vorlesung Programmierparadigmen

- Fokus auf deklarativem Programmieren
  - Funktional: Haskell
  - Logisch: Prolog
- Begleitende Übungsaufgaben:
  - Einreichungen automatisch überprüfen
    - Korrektheit
    - Stil / Technik
  - Hilfestellung durch Feedback während Bearbeitung
- Wir betrachten im Folgenden unser Vorgehen für Haskell-Programmieraufgaben

Haskell-Programmieraufgabe

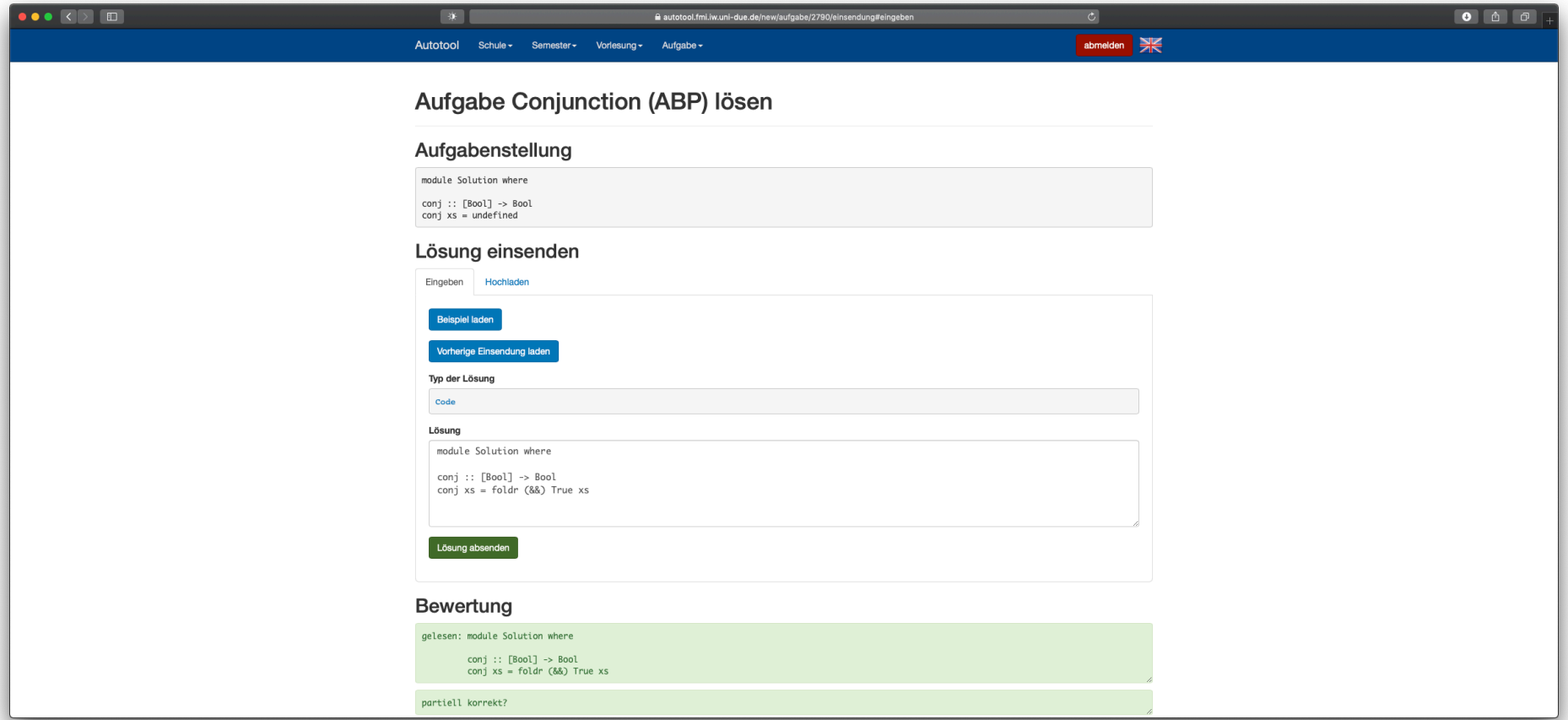
- Schreiben Sie eine Haskell-Funktion

```
con :: [Bool] -> Bool

-- Beispieleingaben:
-- con [False] == False
-- con [True, False] == False
-- con [True, True, True] == True
```

die die Konjunktion aller in einer gegebenen Liste enthaltenen Boolschen Werte berechnet.

Autotool


autotool.fmi.iw.uni-due.de/new/aufgabe/2790/einsendung#eingeben
Autotool
Schule
Semester
Vorlesung
Aufgabe
abmelden
Aufgabe Conjunction (ABP) lösen
Aufgabenstellung
module Solution where
conj :: [Bool] -> Bool
conj xs = undefined
Lösung einsenden
Eingeben
Hochladen
Beispiel laden
Vorherige Einsendung laden
Typ der Lösung
Code
Lösung
module Solution where
conj :: [Bool] -> Bool
conj xs = foldr (&&) True xs
Lösung absenden
Bewertung
gelesen: module Solution where
conj :: [Bool] -> Bool
conj xs = foldr (&&) True xs
partiell korrekt?

Beispielaufgabe: Konjunktion einer Bool-Liste

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Aufgabe

```
module Solution where


con :: [Bool] -> Bool
con = undefined
```

Abgabe

Bewertung

Vorgehen:

- Verwendung mehrerer, etablierter Tools mit an die jeweilige Aufgabe angepassten Konfigurationen

Beispielaufgabe: Konjunktion einer Bool-Liste

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Aufgabe

```
module Solution where


con :: [Bool] -> Bool
con = undefined
```

Abgabe

```
module Solution where


con :: [Bool] -> Bool
con []     = True
con (x:xs) = con xs
```

Bewertung

```
### Failure:
On test case: con [False] == False
```

Vorgehen:

- Verwendung mehrerer, etablierter Tools mit an die jeweilige Aufgabe angepassten Konfigurationen

Beispielaufgabe: Konjunktion einer Bool-Liste

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Aufgabe

```
module Solution where


con :: [Bool] -> Bool
con = undefined
```

Abgabe

```
module Solution where


con :: [Bool] -> Bool
con []     = True
con (x:xs) = x && con xs
```

Bewertung

```
Bewertung der Einsendung: Okay
```

Vorgehen:

- Verwendung mehrerer, etablierter Tools mit an die jeweilige Aufgabe angepassten Konfigurationen

Beispielaufgabe: Konjunktion einer Bool-Liste

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Aufgabe / Abgabe

```
module Solution where


con :: [Bool] -> Bool
con = undefined
```

Potenzielle Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

Bewertung

Vorgehen:

- Verwendung mehrerer, etablierter Tools mit an die jeweilige Aufgabe angepassten Konfigurationen

Beispielaufgabe: Konjunktion einer Bool-Liste

Aufgabe / Abgabe

```
module Solution where


con :: [Bool] -> Bool
con xs = foldr (&&) True xs
```

Potenzielle Lösungen

```
con []     = True
con (x:xs) = x && con xs
-------------------------------------
con xs = foldr (&&) True xs
```

Bewertung

Vorgehen:

- Verwendung mehrerer, etablierter Tools mit an die jeweilige Aufgabe angepassten Konfigurationen

Beispielaufgabe: Konjunktion einer Bool-Liste

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Aufgabe / Abgabe

```
module Solution where


con :: [Bool] -> Bool
con xs = foldr (&&) True xs
```

Potenzielle Lösungen

```
con []     = True
con (x:xs) = x && con xs
-------------------------------------
con xs = foldr (&&) True xs
```

Bewertung

```
Bewertung der Einsendung: Okay
```

Vorgehen:

- Verwendung mehrerer, etablierter Tools mit an die jeweilige Aufgabe angepassten Konfigurationen

GHC – Kompilation

Aufgabe / Abgabe

```
module Solution where


con :: [Bool] -> Bool
con = undefined
```

Potenzielle Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

Bewertung

Kompilermeldungen:

- Syntaxfehler

Aufgabe / Abgabe

```
module Solution where


con :: [Bool] -> Bool
con = if True then
```

Potenzielle Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

Bewertung

Kompilermeldungen:

- Syntaxfehler

Aufgabe / Abgabe

```
module Solution where


con :: [Bool] -> Bool
con = if True then
```

Potenzielle Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

Bewertung

```
Solution.hs:5:1: error:
    parse error (possibly incorrect
    indentation or mismatched brackets)
```

Kompilermeldungen:

- Syntaxfehler

Aufgabe / Abgabe

```
module Solution where


con :: [Bool] -> Bool
con = undefined
```

Potenzielle Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

Bewertung

Kompilermeldungen:

- Syntaxfehler
- Typfehler

Aufgabe / Abgabe

```
module Solution where


con :: [Bool] -> Bool
con xs = length xs
```

Potenzielle Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

Bewertung

Kompilermeldungen:

- Syntaxfehler
- Typfehler

Aufgabe / Abgabe

```
module Solution where


con :: [Bool] -> Bool
con xs = length xs
```

Potenzielle Lösungen

```
con []     = True
con (x:xs) = x && con xs
-------------------------------------
con xs = foldr (&&) True xs
```

Bewertung

```
Solution.hs:5:11: error:
    • Couldn't match expected type 'Bool'
      with actual type 'Int'
    • In the expression: length xs
      In an equation for 'con':
      con xs = length xs
```

Kompilermeldungen:

- Syntaxfehler
- Typfehler

GHC – Kompilation

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Aufgabe / Abgabe

```
module Solution where


con :: [Bool] -> Bool
con = undefined
```

In der Konfiguration

```
configGhcErrors:
 - incomplete-patterns
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

Bewertung

Kompilermeldungen:

- Syntaxfehler
- Typfehler

Weitere statische Analysen:

- Erkennen unvollständiger Fallunterscheidungen

GHC – Kompilation

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Aufgabe / Abgabe

```
module Solution where


con :: [Bool] -> Bool
con []     = True
```

In der Konfiguration

```
configGhcErrors:
 - incomplete-patterns
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

Bewertung

Kompilermeldungen:

- Syntaxfehler
- Typfehler

Weitere statische Analysen:

- Erkennen unvollständiger Fallunterscheidungen

GHC – Kompilation

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Aufgabe / Abgabe

```
module Solution where


con :: [Bool] -> Bool
con []     = True
```

In der Konfiguration

```
configGhcErrors:
 - incomplete-patterns
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

Bewertung

```
Solution.hs:5:1: error:
    Pattern match(es) are non-exhaustive
    In an equation for ‘con’:
    Patterns not matched: (_:_)
```

Kompilermeldungen:

- Syntaxfehler
- Typfehler

Weitere statische Analysen:

- Erkennen unvollständiger Fallunterscheidungen

GHC – Kompilation

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Aufgabe / Abgabe

```
module Solution where


con :: [Bool] -> Bool
con []     = True
```

In der Konfiguration

```
configGhcErrors:
 - incomplete-patterns
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

Bewertung

```
Solution.hs:5:1: error:
    Pattern match(es) are non-exhaustive
    In an equation for ‘con’:
    Patterns not matched: (_:_)
```

Kompilermeldungen:

- Syntaxfehler
- Typfehler

Weitere statische Analysen:

- Erkennen unvollständiger Fallunterscheidungen

GHC – Kompilation

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Aufgabe / Abgabe

```
module Solution where


con :: [Bool] -> Bool
con []     = True
con (x:xs) = x && con xs
```

In der Konfiguration

```
configGhcErrors:
 - incomplete-patterns
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
-----------------------------
con xs = foldr (&&) True xs
```

Bewertung

```
Solution.hs:5:1: error:
    Pattern match(es) are non-exhaustive
    In an equation for ‘con’:
    Patterns not matched: (_:_)
```

Kompilermeldungen:

- Syntaxfehler
- Typfehler

Weitere statische Analysen:

- Erkennen unvollständiger Fallunterscheidungen

GHC – Kompilation

Aufgabe / Abgabe

```
module Solution where


con :: [Bool] -> Bool
con []     = True
con (x:xs) = x && con xs
```

In der Konfiguration

```
configGhcErrors:
 - incomplete-patterns
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

Bewertung

```
Bewertung der Einsendung: Okay
```

Kompilermeldungen:

- Syntaxfehler
- Typfehler

Weitere statische Analysen:

- Erkennen unvollständiger Fallunterscheidungen

GHC – Kompilation

Aufgabe / Abgabe

```
module Solution where


con :: [Bool] -> Bool
con []     = True
con (x:xs) = x && con xs
```

In der Konfiguration

```
configGhcErrors:
 - incomplete-patterns
configGhcWarnings:
 - name-shadowing
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
-------------------------
con xs = foldr (&&) True xs
```

Bewertung

```
Bewertung der Einsendung: Okay
```

Kompilermeldungen:

- Syntaxfehler
- Typfehler

Weitere statische Analysen:

- Erkennen unvollständiger Fallunterscheidungen
- Stilvorgaben

GHC – Modulsystem

Aufgabe / Abgabe

```
module Solution where


con :: [Bool] -> Bool
con = undefined
```

In der Konfiguration

```
configGhcErrors:
 - incomplete-patterns
configGhcWarnings:
 - name-shadowing
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
-----------------------------
con xs = foldr (&&) True xs
```

Bewertung

Auswahl verfügbarer Funktionen durch selektive Imports:

- Vordefinierte Funktionen ausschließen
- Einschränkungen basierend auf Kursfortschritt

Aufgabe / Abgabe

```
module Solution where


con :: [Bool] -> Bool
con = undefined
```

In der Konfiguration

```
configGhcErrors:
 - incomplete-patterns
configGhcWarnings:
 - name-shadowing
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

```
con xs = and xs
```

Bewertung

Auswahl verfügbarer Funktionen durch selektive Imports:

- Vordefinierte Funktionen ausschließen
- Einschränkungen basierend auf Kursfortschritt

GHC – Modulsystem

Aufgabe / Abgabe

```
module Solution where
import Prelude hiding (and)

con :: [Bool] -> Bool
con = undefined
```

In der Konfiguration

```
configGhcErrors:
 - incomplete-patterns
configGhcWarnings:
 - name-shadowing
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

```
con xs = and xs
```

Bewertung

Auswahl verfügbarer Funktionen durch selektive Imports:

- Vordefinierte Funktionen ausschließen
- Einschränkungen basierend auf Kursfortschritt

GHC – Modulsystem

Aufgabe / Abgabe

```
module Solution where
import Prelude hiding (and,
  foldr)
con :: [Bool] -> Bool
con = undefined
```

In der Konfiguration

```
configGhcErrors:
 - incomplete-patterns
configGhcWarnings:
 - name-shadowing
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

```
con xs = and xs
```

Bewertung

Auswahl verfügbarer Funktionen durch selektive Imports:

- Vordefinierte Funktionen ausschließen
- Einschränkungen basierend auf Kursfortschritt

GHC – Modulsystem

Aufgabe / Abgabe

```
module Solution where
import Prelude hiding (and)

con :: [Bool] -> Bool
con xs = and xs
```

In der Konfiguration

```
configGhcErrors:
 - incomplete-patterns
configGhcWarnings:
 - name-shadowing
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

```
con xs = and xs
```

Bewertung

Auswahl verfügbarer Funktionen durch selektive Imports:

- Vordefinierte Funktionen ausschließen
- Einschränkungen basierend auf Kursfortschritt

Aufgabe / Abgabe

```
module Solution where
import Prelude hiding (and)

con :: [Bool] -> Bool
con xs = and xs
```

In der Konfiguration

```
configGhcErrors:
 - incomplete-patterns
configGhcWarnings:
 - name-shadowing
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

```
con xs = and xs
```

Bewertung

```
Solution.hs:6:8: error:
    • Variable not in scope: and :: [Bool] -> Bool
    • Perhaps you meant one of these:
        ‘any’ (imported from Prelude), ‘snd’ (imported from
        Prelude)
      Perhaps you want to remove ‘and’ from the explicit
      hiding list
      in the import of ‘Prelude’ (Solution.hs:3:1-27).
```

Auswahl verfügbarer Funktionen durch selektive Imports:

- Vordefinierte Funktionen ausschließen
- Einschränkungen basierend auf Kursfortschritt

HLint – Konfigurierbarer Linter

HLint – Konfigurierbarer Linter

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Aufgabe / Abgabe

```
module Solution where
import Prelude hiding (and)

con :: [Bool] -> Bool
con = undefined
```

In der Konfiguration

```
configGhcErrors:
 - incomplete-patterns
configGhcWarnings:
 - name-shadowing
configHlintErrors:
 - Redundant ==
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

```
con xs = and xs
```

Bewertung

Vorschläge zur Refaktorisierung:

- Aufzeigen idiomatischen Vorgehens
- Vermeiden umständlicher Lösungen
  - Hervorhebung spezieller Haskell-Features

Konfiguration dieser Vorschläge als Fehler

- Forcierung der Umsetzung
- Führt zur Normalisierung studentischer Einreichungen

HLint – Konfigurierbarer Linter

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Aufgabe / Abgabe

```
module Solution where
import Prelude hiding (and)

con :: [Bool] -> Bool
con []     = True
con (x:xs) =
  (x == True) && con xs
```

In der Konfiguration

```
configGhcErrors:
 - incomplete-patterns
configGhcWarnings:
 - name-shadowing
configHlintErrors:
 - Redundant ==
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

```
con xs = and xs
```

Bewertung

Vorschläge zur Refaktorisierung:

- Aufzeigen idiomatischen Vorgehens
- Vermeiden umständlicher Lösungen
  - Hervorhebung spezieller Haskell-Features

Konfiguration dieser Vorschläge als Fehler

- Forcierung der Umsetzung
- Führt zur Normalisierung studentischer Einreichungen

HLint – Konfigurierbarer Linter

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Aufgabe / Abgabe

```
module Solution where
import Prelude hiding (and)

con :: [Bool] -> Bool
con []     = True
con (x:xs) =
  (x == True) && con xs
```

In der Konfiguration

```
configGhcErrors:
 - incomplete-patterns
configGhcWarnings:
 - name-shadowing
configHlintErrors:
 - Redundant ==
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

```
con xs = and xs
```

Bewertung

```
7:16: Warning: Redundant ==
Found:
  x == True
Perhaps:
  x
```

Vorschläge zur Refaktorisierung:

- Aufzeigen idiomatischen Vorgehens
- Vermeiden umständlicher Lösungen
  - Hervorhebung spezieller Haskell-Features

Konfiguration dieser Vorschläge als Fehler

- Forcierung der Umsetzung
- Führt zur Normalisierung studentischer Einreichungen

HLint – Konfigurierbarer Linter

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Aufgabe / Abgabe

```
module Solution where
import Prelude hiding (and)

con :: [Bool] -> Bool
con = undefined
```

In der Konfiguration

```
configGhcErrors:
 - incomplete-patterns
configGhcWarnings:
 - name-shadowing
configHlintErrors:
 - Redundant ==
 - Redundant if
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

```
con xs = and xs
```

Bewertung

Vorschläge zur Refaktorisierung:

- Aufzeigen idiomatischen Vorgehens
- Vermeiden umständlicher Lösungen
  - Hervorhebung spezieller Haskell-Features

Konfiguration dieser Vorschläge als Fehler

- Forcierung der Umsetzung
- Führt zur Normalisierung studentischer Einreichungen

HLint – Konfigurierbarer Linter

Aufgabe / Abgabe

```
module Solution where
import Prelude hiding (and)

con :: [Bool] -> Bool
con []     = True
con (x:xs) =
 if x then con xs else False
```

In der Konfiguration

```
configGhcErrors:
 - incomplete-patterns
configGhcWarnings:
 - name-shadowing
configHlintErrors:
 - Redundant ==
 - Redundant if
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

```
con xs = and xs
```

Bewertung

Vorschläge zur Refaktorisierung:

- Aufzeigen idiomatischen Vorgehens
- Vermeiden umständlicher Lösungen
  - Hervorhebung spezieller Haskell-Features

Konfiguration dieser Vorschläge als Fehler

- Forcierung der Umsetzung
- Führt zur Normalisierung studentischer Einreichungen

HLint – Konfigurierbarer Linter

Aufgabe / Abgabe

```
module Solution where
import Prelude hiding (and)

con :: [Bool] -> Bool
con []     = True
con (x:xs) =
 if x then con xs else False
```

In der Konfiguration

```
configGhcErrors:
 - incomplete-patterns
configGhcWarnings:
 - name-shadowing
configHlintErrors:
 - Redundant ==
 - Redundant if
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

```
con xs = and xs
```

Bewertung

```
7:15: Warning: Redundant if
Found:
  if x then con xs else False
Perhaps:
  x && con xs
```

Vorschläge zur Refaktorisierung:

- Aufzeigen idiomatischen Vorgehens
- Vermeiden umständlicher Lösungen
  - Hervorhebung spezieller Haskell-Features

Konfiguration dieser Vorschläge als Fehler

- Forcierung der Umsetzung
- Führt zur Normalisierung studentischer Einreichungen


UNIVERSITÄT
DUISBURG
ESSEN
Offen im Denken
Mustervergleich

Mustervergleich

Aufgabe / Abgabe

```
module Solution where
import Prelude hiding (and)

con :: [Bool] -> Bool
con xs = foldr
    undefined undefined xs
```

In der Konfiguration

```
configGhcErrors:
 - incomplete-patterns
configGhcWarnings:
 - name-shadowing
configHlintErrors:
 - Redundant ==
 - Redundant if
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

```
con xs = and xs
```

Bewertung

- Vorgabe der allgemeinen Lösungsstruktur
- Steuerung der Lösungsstrategie durch stärkere Vorgaben

Mustervergleich

Aufgabe / Abgabe

```
module Solution where
import Prelude hiding (and)

con :: [Bool] -> Bool
con xs = foldr
    undefined undefined xs
```

In der Konfiguration

```
configGhcErrors:
 - incomplete-patterns
configGhcWarnings:
 - name-shadowing
configHlintErrors:
 - Redundant ==
 - Redundant if
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

```
con xs = and xs
```

Bewertung

- Vorgabe der allgemeinen Lösungsstruktur
- Steuerung der Lösungsstrategie durch stärkere Vorgaben

Mustervergleich

Aufgabe / Abgabe

```
module Solution where



con :: [Bool] -> Bool
con xs = and xs
con xs = foldr
    undefined undefined xs
```

In der Konfiguration

```
configGhcErrors:
 - incomplete-patterns
configGhcWarnings:
 - name-shadowing
configHlintErrors:
 - Redundant ==
 - Redundant if
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

```
con xs = and xs
```

Bewertung

- Vorgabe der allgemeinen Lösungsstruktur
- Steuerung der Lösungsstrategie durch stärkere Vorgaben

Mustervergleich

Aufgabe / Abgabe

```
module Solution where


con :: [Bool] -> Bool
con xs = and xs
con xs = foldr
    undefined undefined xs
```

In der Konfiguration

```
configGhcErrors:
 - incomplete-patterns
configGhcWarnings:
 - name-shadowing
configHlintErrors:
 - Redundant ==
 - Redundant if
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

```
con xs = and xs
```

Bewertung

```
Your solution does not match the template:
Template:                                  Submission:
+----------------------------              +----------------------------
|                                          |
| con :: [Bool] -> Bool                    | con :: [Bool] -> Bool
| con xs = foldr undefined undefined xs    | con xs = and xs
|          ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^  |          ^^^^^^^
+----------------------------              +----------------------------
```

- Vorgabe der allgemeinen Lösungsstruktur
- Lösungsstrategie durch
  ben


UNIVERSITÄT
DUISBURG
ESSEN
Offen im Denken
QuickCheck

Aufgabe / Abgabe

```
module Solution where
import Prelude hiding (and)

con :: [Bool] -> Bool
con xs = foldr
    undefined undefined xs
```

Test-Suite

```
singletonList :: Bool -> Bool
singletonList x =
     con [x] == x
concatenatedLists
  :: [Bool] -> [Bool] -> Bool
concatenatedLists xs ys =
     con (    xs ++     ys)
  ==     (con xs && con ys)
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

```
con xs = and xs
```

Bewertung

Testen der Einreichung:

- Eigenschaftsbasiertes Testen
- Definition von Eigenschaften durch Haskell-Funktionen
- Ausführung der Eigenschaften auf zufällig erzeugten Eingaben (durch QuickCheck)

Aufgabe / Abgabe

```
module Solution where
import Prelude hiding (and)

con :: [Bool] -> Bool
con xs = foldr
         (||)   False   xs
```

Test-Suite

```
singletonList :: Bool -> Bool
singletonList x =
     con [x] == x
concatenatedLists
  :: [Bool] -> [Bool] -> Bool
concatenatedLists xs ys =
     con (    xs ++     ys)
  ==     (con xs && con ys)
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

```
con xs = and xs
```

Bewertung

Testen der Einreichung:

- Eigenschaftsbasiertes Testen
- Definition von Eigenschaften durch Haskell-Funktionen
- Ausführung der Eigenschaften auf zufällig erzeugten Eingaben (durch QuickCheck)

Aufgabe / Abgabe

```
module Solution where
import Prelude hiding (and)

con :: [Bool] -> Bool
con xs = foldr
         (||)   False   xs
```

Test-Suite

```
singletonList :: Bool -> Bool
singletonList x =
     con [x] == x
concatenatedLists
  :: [Bool] -> [Bool] -> Bool
concatenatedLists xs ys =
     con (    xs ++     ys)
  ==     (con xs && con ys)
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

```
con xs = and xs
```

Bewertung

```
### Failure in: concatenatedLists
*** Failed! Falsifiable (after 2 tests):
xs = [True]
ys = []
```

Testen der Einreichung:

- Eigenschaftsbasiertes Testen
- Definition von Eigenschaften durch Haskell-Funktionen
- Ausführung der Eigenschaften auf zufällig erzeugten Eingaben (durch QuickCheck)

UNIVERSITÄT DUISBURG ESSEN

Offen im Denken

Zusammenfassung

Zusammenfassung

Aufgabe / Abgabe

```
module Solution where
import Prelude hiding (and)

con :: [Bool] -> Bool
con xs = foldr (&&) True xs
```

In der Konfiguration

```
configGhcErrors:
 - incomplete-patterns
configGhcWarnings:
 - name-shadowing
configHlintErrors:
 - Redundant ==
 - Redundant if
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

```
con xs = and xs
```

Bewertung

```
Bewertung der Einsendung: Okay
```

- Kombination etablierter Haskell Tools
- Reihenfolge auf schrittweises Lösen von Aufgaben abgestimmt
- Detaillierte Konfiguration erlaubt Anpassung an konkrete Aufgabenstellung
- Überprüfung von
  - funktionaler Korrektheit
  - stilistischen Vorgaben
  - Verwendung bestimmter Programmierkonzepte

Ausblick

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Weitere Features

- Gestaffelte Ausführung von Tests für detaillierteres Feedback
  (z. B. konkrete Beispiele aus Aufgabenstellung vor allgemeinen Tests)
- Testen interaktiven Verhaltens von IO-Programmen
- Analyse des Syntaxbaumes
  (z.B. Anzahl der verwendeten Fallunterscheidungen bzw. Hilfsfunktionen überprüfen)
- Geschickte Kombination verschiedener Tools zur Umsetzung komplexerer Anforderungen

Erweiterungsmöglichkeiten

- Detaillierteres Feedback und Fehlermeldungen
- Randomisierung / automatische Aufgabenerstellung
  (für bestimmte Aufgabentypen)
- Testen von Haskellaufgaben zur Erzeugung von einfachen Grafiken (CodeWorld)

Zusammenfassung

Aufgabe / Abgabe

```
module Solution where
import Prelude hiding (and)

con :: [Bool] -> Bool
con  xs = foldr (&&) True xs
```

In der Konfiguration

```
configGhcErrors:
 - incomplete-patterns
configGhcWarnings:
 - name-shadowing
configHlintErrors:
 - Redundant ==
 - Redundant if
```

Potenz. Lösungen

```
con []     = True
con (x:xs) = x && con xs
```

```
con xs = foldr (&&) True xs
```

```
con xs = and xs
```

Bewertung

```
Bewertung der Einsendung: Okay
```

- Kombination etablierter Haskell Tools
- Reihenfolge auf schrittweises Lösen von Aufgaben abgestimmt
- Detaillierte Konfiguration erlaubt Anpassung an konkrete Aufgabenstellung
- Überprüfung von
  - funktionaler Korrektheit
  - stilistischen Vorgaben
  - Verwendung bestimmter Programmierkonzepte