Specyfikacje formalne i programy funkcyjne - zadanie projektowe z języka Haskell

Antonina Lobach
Gabriela Ossowska

Algorytm poszukiwania rozwiązania:

Poruszając się po elementach planszy po zmianie statusu komórki na “Filled” albo “Empty”, sprawdza się wpływ zmiany statusu na prawidłowość rozwiązania. Algorytm sprawdza możliwe kombinacje podstawienia statusu do zwrócenia poprawnego rozwiązania.

Pliki rozwiązania:

Solver.hs - zawiera funkcję *main*, gdzie przy użyciu funkcji *readPuzzle* mozaika jest wczytywana z pliku, sparsowana do zmiennej typu *Board* i przekazana jako argument do funkcji *solve*, która drukuje rozwiązanie.

Field.hs - definicja komórki/pola łamigłówki. Field takie własności jak Value - wartość maybe Int, State: Filled (zamalowane), Empty (niezamalowane), Null (nie ustawione). Funkcje:

- *toDefaultField* - funkcja do serializacji wartości z pliku sprasowanego,
- *getFieldPrint* - zwraca znak statusu do wypisania do konsoli.

Board.hs - mozaika jest reprezentowana w programie jako zmienna typu *Board* zdefiniowana jako *[[Field]]*. Funkcje:

- *getIndexies* i *printBoardIndexies* służą do stworzenia i wydrukowania listy indeksów elementów mozaiki,
- *getField* zwraca pole dla podanych indeksów,
- *parseToBoard* wczytuje mozaikę, *printBoard* drukuje mozaikę,
- *changeState* - zmienia status pola o zadanych współrzędnych na wybrany - zamalowane, puste, null. Przechodzi rekurencyjnie po wierszach mozaiki, do przechodzenia po polach wiersza używa *checkCell*,
- *checkCell* - przyjmuje listę pól i zmienia status wskazanego pola na zadany,
- *cutCellGroup* - dla pola mozaiki o wskazanym indeksie wycina taki jej fragment, który zawiera tylko wskazane pole i wszystkie te, które się z nim stykają,
- *countStateForClue* - sprawdza dla wskazanego pola, ile pól ze zbioru zwróconego dla niego przez *cutCellGroup* ma zadany status - ile z nich jest zamalowanych, pustych albo o statusie *null*,
- *collectCluesAroundCell* - zwraca listę opisującą podpowiedzi wokół pola o zadanych współrzędnych; wycina fragment wokół pola (używa *cutCellGroup*), szuka w nim pól z podpowiedziami i zapisuje je w liście razem z jego współrzędnymi. Lokalna funkcja *collectClues* przechodzi przez fragment wiersz po wierszu, *collectCluesFromRow* przyjmuje wiersz i przechodzi po nim pole po polu,
- *checkIfEqual*, *checkIfNotMore* - służą do sprawdzenia, czy wokół podpowiedzi zostało zamalowanych tyle pól, ile ona określa (albo nie więcej, niż ona określa)
- *checkValidityAroundPosition* - służy do sprawdzenia, jak ustalenie statusu kolejnego pola (zamalowanie/zostawienie pustego) wpłynęło na prawidłowość rozwiązania; dla każdej podpowiedzi z listy zwracanej przez *collectCluesAroundCell* sprawdza lokalną funkcją *checkCluesValidity*, czy wokół niej zamalowanych jest tyle pól, ile wskazuje, albo przynajmniej nie więcej (jeżeli nie wszystkie pola wokół niej mają już ustalone statusy),
- *solve* - rekurencyjnie generuje możliwe rozwiązania, odrzuca błędne, drukuje prawidłowe.