## Робота №7

Бінарні діаграми рішень

*Булеві вирази* включають булеві константи *True* і *False*, булеві змінні (тобто *x1, x2,* …), заперечення (¬) і бінарні операції (Ʌ) і (V), котрі мають звичайну семантику (Таб.1)

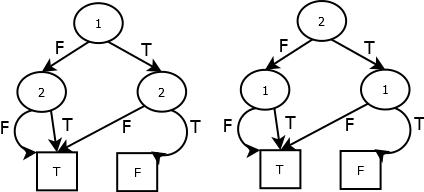
|  |  |  |  |
| --- | --- | --- | --- |
| *a* | *b* | *a Ʌ b* | *a V b* |
| *False* | *False* | *False* | *False* |
| *False* | *True* | *False* | *True* |
| *True* | *False* | *False* | *True* |
| *True* | *True* | *True* | *True* |

|  |  |
| --- | --- |
| *a* | *¬a* |
| *False* | *True* |
| *True* | *False* |

*Таб.1 Таблиці істини (здійсненності) для ¬, Ʌ і V*

Булевий вираз можна представити у вигляді бінарної діаграми рішення. *Бінарна діаграма рішення (БДР)* - це орієнтованого ациклічного графа, котрий дуже схожий на бінарне дерево, що кодує всі можливі значення булевих змінних, що входять у вираз. Граф містить внутрішні вузли, що позначаються цілими – індекси змінних і мають правого і лівого сина, лівий син завжди відповідає *False* і правий син *True*. Кожний шлях в графі представляє унікальний набір значень всіх змінних виразу і веде від кореня до вузла листка, котрий представляє *False* або *True –* значення булевого виразу при відповідному наборі значень змінних. Якщо різні змінні з’являються в тому ж порядку на всіх шляхах від кореня, то бінарна діаграма рішення – впорядкована.

Мал.1 показує дві впорядковані БДР для виразу ¬(x1 Ʌ (False V x2 )) : зліва - змінні з’являються в порядку [1, 2] зверху до низу , справа - змінні з’являються в порядку [2, 1].

**Листки F і T - це скорочення для *False* і *True,* номер зв’язаний з кожним внутрішнім вузлом є індекс змінної.

*Мал.1.* *Два повні БДР для ¬(x1 Ʌ (False V x2 ))*

Булеві вирази і БДР в Haskell можна задати наступними типами даних

.

***type*** Index = Int

***data*** BExp = Prim Bool | IdRef Index | Not BExp | And BExp BExp | Or BExp BExp

deriving (Eq, Ord, Show)

***type*** Env = [(Index, Bool)]

***type*** NodeId = Int

***type*** BDDNode = (NodeId, (Index, NodeId, NodeId))

***type*** BDD = (NodeId, [BDDNode])

Булевий вираз *¬ (x1 Ʌ (False V x2 ))* можна представити так:

Not (And (IdRef 1) (Or (Prim False) (IdRef 2)))

Дані типу Env використовуються для задання конкретного набору значень булевих змінних

Ідентифікатори вузлів (NodeId) в БДР повинні бути унікальними. Вузли листки явно не включаються в представлення БДР. Замісто цього, резервуються мітки 0 і 1, котрі будуть використовуватися для «невидимих», але завжди-присутніх *False* і *True* відповідно.

БДР на Мал. 1 можна відповідно представити:

(2, [(4, (2,1,1)), (5, (2,1,0)), (2, (1,4,5))])

(2 ,[(2, (2,7,3)), (7, (1,1,1)), (3, (1,1,0))])

Головна ідея побудови за виразом БДР базується на розкладенні Шеннона. Будь який булевий вираз по одній із змінних можна розділити на два підвирази (підставляючи у вираз значення *False* і *True* відповідно і, можливо, скорочуючи результат). Представляючи кожний такий вираз у вигляді піддерева і продовжуючи розкладання по решті вхідних змінних можна отримати БДР.

Побудувати БДР за булевим виразом *e* з урахуванням порядку змінних, що задається списком *xs*, може наступна функція *buildBDD e xs*, що використовує допоміжну функцію *buildBDD’ e 2 xs.*

buildBDD :: BExp -> [Index] -> BDD

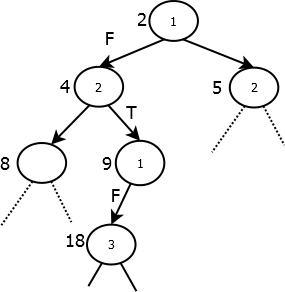
buildBDD e xs = buildBDD’ e 2 xs

buildBDD’ :: BExp -> NodeId -> [Index] -> BDD

buildBDD’ = undefined

NodeId – це ідентифікатор, котрий потрібно присвоювати наступному вузлу в БДР, що будуєте. Тому що ідентифікатори вузлів 0 і 1 вже зарезервовані для представлення False і

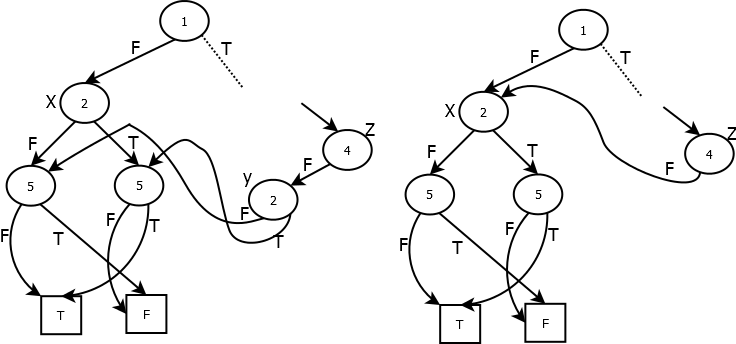
True відповідно, то перший невикористаний ідентифікатор є 2.

На кожному кроці побудови із списку змінних *xs* вибирається перша, по якій вираз *e* ділиться на два підвирази (розкладення Шеннона), за якими рекурсивно будуються ліве і праве піддерева. Коли список *xs* порожній ([]), то досягли листка дерева. В цьому випадку повертаєте БДР виду (b,[]), де b або 0 або 1, котра представляє *False* або *True* відповідно.

*Мал.2: Побудова БДР*

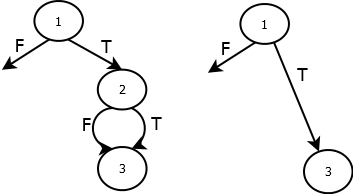
При побудови БДР потрібно щоб кожний вузол має унікальний ідентифікатор. Один із способів: якщо вузол помічений n, то вузли корені лівого та правого піддерев помічаються 2n і 2n+1 відповідно ( Мал. 2).

Повна впорядкована БДР, що будується за таким алгоритмом має структуру, котру можна суттєво скоротити і отримати *скорочену впорядковану БДР.* Скорочена БДР отримується в результаті застосування до повної БДР двох правил:

* Злиті вузли-дублікати («злиті будь які ізоморфні підграфи»).
  + Якщо два вузли (m1,(i1,f1,t1)) і (m2,(i2,f2,t2)) відрізняються лише мітками, тобто m1≠m2, i1=i2, f1=f2, t1=t2, то один з цих вузлів (наприклад - m2) можна вилучити, а всі посилання на нього спрямувати на інший вузол (m1).
* Вилучені лишні перевірки («вилучено будь який вузол, у якого обидва нащадки ізоморфні»).
  + Якщо у вузла (m,(i,f,t)), f= t, то його можна вилучити, а всі посилання на нього спрямувати на f.

*Мал.3: Злиття вузлів-дублікатів («Злиття ізоморфних підграфів»)*

Перше – злиття вузлів-дублікатів («спільне використання підграфів») всюди де це можливо. Приклад показано на Мал.3. Тут вузол, що помічено як X, ідентичний вузлу поміченому як Y. Вузол Y можна вилучити і всі посилання на нього спрямувати на X.

Друге – вилучити лишню перевірку, тобто вузол який посилається двічі на одне і те ж дерево, як показано на Мал.4

*Мал.4. Вилучення лишньої перевірки («У вузла обидва нащадки ізоморфні»).*

Розмір БДР визначається як булевим виразом, так і вибором порядку змінних. На Мал.5 повна впорядкована БДР для виразу (x1 Ʌ x2) V (x3 Ʌ x4). На Мал.6 дві скорочені БДР для того ж виразу, що отриманні використовуючи порядок індексів змінних [1, 3, 2, 4] і [1, 2, 3, 4].

В допоміжному файлі, котрий включає визначення типів, допоміжних функцій і даних для тестування, надати визначення наступних функцій.

1. Функція *checkSat bdd env*, котра повертає True якщо значення змінних задані в середовищі *env* задовольняють вираз, що відповідає БДР *bdd*. Наприклад:

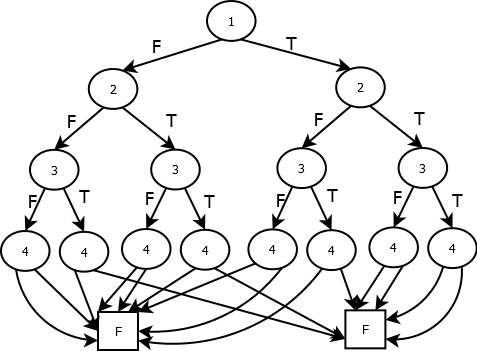
checkSat bdd2 [(1,True),(2,False)] = True

checkSat bdd7 [(3,True),(2,False),(9,True)]= False

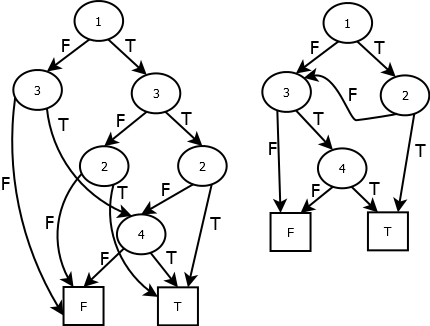
1. Функція *sat bdd,* котра обчислює список всіх множин значень змінних, котрі задовольняють булевий вираз, що відповідає БДР *bdd*. Кожна «множина» значень є список пар (Index, Bool). Наприклад:

sat bdd1 = []

sat bdd2 = [[(1,False),(2,False)],[(1,False),(2,True)],[(1,True),(2,False)]]

sat bdd8 = [[(1,False)],[(1,True)]] 

*Мал.5: Повна БДР для (x1 Ʌ x2) V (x3 Ʌ x4)*



*Мал.6: Скорочені БДР для (x1 Ʌ x2) V (x3 Ʌ x4)*

1. Нерекурсивна функція *simplify e*, що спрощує булевий вираз *e*, застосовуючи правила визначені в таблицях істини (Таб.1) у випадках коли аргумент (и) ¬, Ʌ і V є булеві константи (Prim False або Prim True). Якщо таких немає, то вираз повертається незмінним . Наприклад:

simplify (Not (Prim False)) = Prim True

simplify (Or (Prim False) (Prim False)) = Prim False

simplify (And (IdRef 3) (Prim True)) = And (IdRef 3) (Prim True)

1. Функція *restrict e i v*, котра замінює вказану змінну (індекс *i*) у виразі *e*, на булеву константу *v*. Заміна може дозволити застосувати одно або більше правил спрощення, тому потрібно викликати *simplify* кожний раз, коли будується новий Not, And або Or вираз. Наприклад.

restrict b7 2 True = Or (Not (IdRef 3)) (Or (Prim True) (Not (IdRef 9)))

restrict (restrict b7 2 True) 9 False = Or (Not (IdRef 3)) (Prim True)

restrict (restrict (restrict b7 2 True) 9 False) 3 True = Prim True

1. Допоміжна функція *builBDD’ e n xs*, котра будує БДР за булевим виразом *e* з урахуванням порядку змінних, що задається списком *xs*; *n -* це ідентифікатор наступного вузла в БДР, що будується.Наприклад (з точністю до ідентифікаторів вузлів і порядку елементів у списку):

buildBDD b1 [] = bdd1

buildBDD’ b2 2 [1, 2] = bdd2

buildBDD b4 [2,3,7] = bdd4

1. Функція *buildROBDD e xs*, котра будує скорочену впорядковану БДР за булевим виразом *e* з урахуванням порядку змінних, що задається списком *xs,* застосовуючи наведені раніше оптимізації до БДР, що будується функцією buildBDD. Наприклад (з точністю до ідентифікаторів вузлів і порядку елементів у списку):

buildROBDD b6 [1,2,3,4] = (5,[(5,(1,3,4)),(3,(3,0,2)),(2,(4,0,1)),(4,(2,3,1))])

buildROBDD b6 [1,3,2,4] = (7,[(7,(1,3,6)),(3,(3,0,2)),(2,(4,0,1)),(6,(3,4,5)),(4,(2,0,1)),(5,(2,2,1))])

checkSat :: BDD -> Env -> Bool

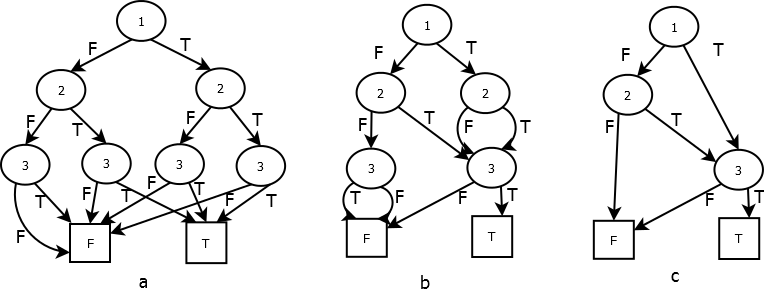
sat :: BDD -> [[(Index, Bool)]]

simplify :: BExpr -> BExpr

restrict :: BExp -> Index -> Bool -> BExpr

buildBDD’ :: BExp -> NodeId -> [Index] -> BDD

buildROBDD :: BExp -> [Index] -> ROBDD



*Мал.7: Побудова скороченої впорядкованої СВБДР для x3 Ʌ (x2 V (¬x2 Ʌ x1)): повна БДР (a), злиття вузлів-дублікатів (b), вилучення лишніх перевірок (c).*

Зауваження:

Назва файлу Family07.hs (Family – прізвище студента). Файл включає модуль Family07 і створюється на основі файла-заготовки HWP07.hs