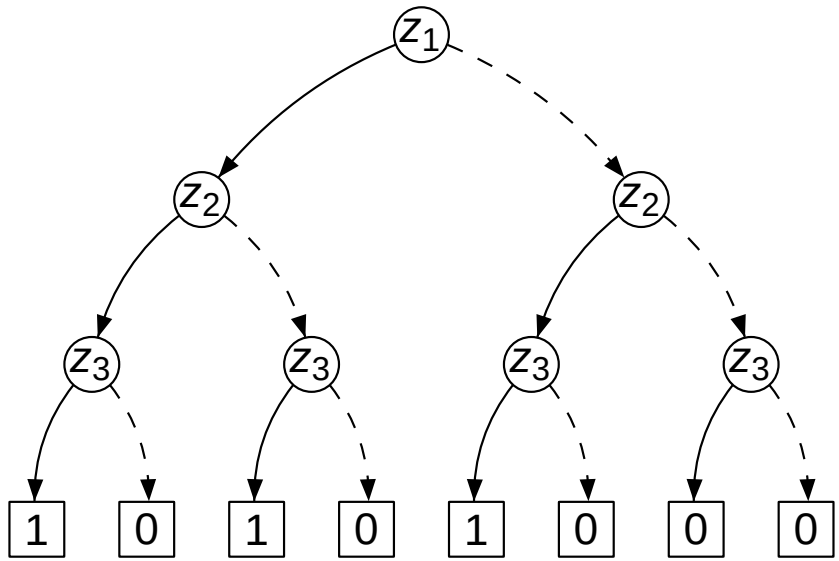
Langages de spécification – cours 4

Diagrammes de décision binaire(BDD)

Catalin Dima

Arbres de décision binaire

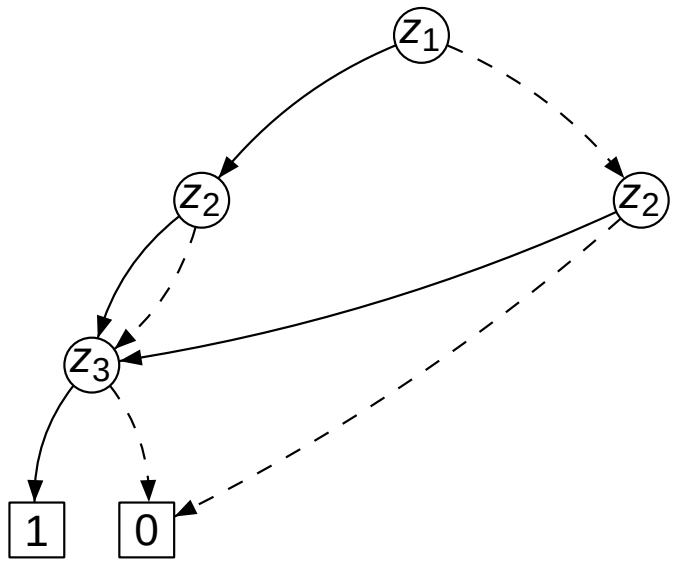
- Étant donnée une formule logique, on peut lui associer un arbre qui permet d'évaluer la valeur de vérité de la formule en fonction des valeurs de vérité des variables.
- Exemple : $f = (z_1 \wedge z_3) \vee (z_2 \wedge z_3)$:



Fusion des nodes

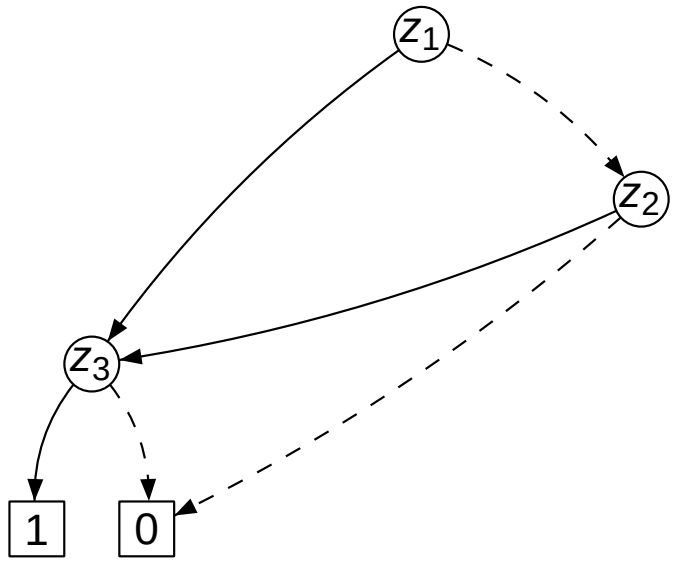
- Un noeud qui n'est pas relié à la racine peut être supprimé.
- Deux noeuds qui ont des sous-arbres gauche et droite identiques peuvent être fusionnés.
- Un noeud peut être supprimé si ses deux successeurs pointent vers le même noeud (noeud inutile).

Résultat intermédiaire pour $f = (z_1 \wedge z_3) \vee (z_2 \wedge z_3)$:



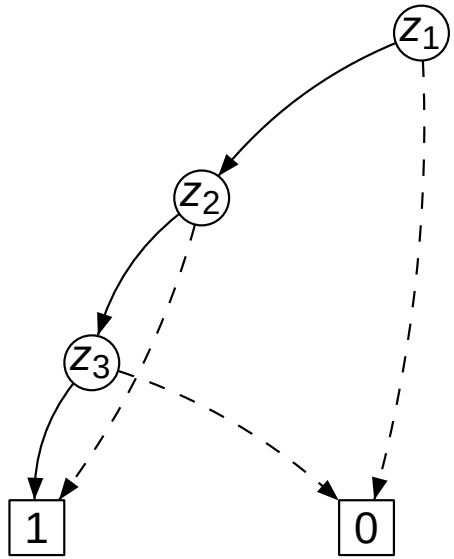
Fusion des noeuds

Résultat final :



Autres exemples

- $f_2 = z_1 \wedge (\neg z_2 \vee z_3)$.



BDD = Binary Decision Diagrams

Définition

Soit $P = \{x_1, \ldots, x_n\}$ un ensemble de variables ordonné (c.à.d. on considère $x1 < x2 < \ldots < xn$). Un diagramme de décision binaire (BDD) est un *graphe dirigé étiqueté acyclique* avec une racine tel que :

- Exactement un ou deux noeuds terminaux (sans successeurs) sont étiquetés par 0 ou 1.
- Les autres noeuds sont étiquetés par des variables de P et ont exactement deux successeurs.
- Tout chemin de la racine aux feuilles respecte l'ordre des variables $x_1 < \ldots < x_n$.

Notons $var(u)$ la variable du noeud u, $low(u)$ et $high(u)$ les deux successeurs d'un tel noeud u

- (si u est étiqueté par x_i , alors $low(u)$ correspond à la valeur 0 pour la variable x_i et $high(u)$ à la valeur 1).

Évaluation d'un BDD

- On parcourt le BDD en partant du ou des noeuds terminaux, en associant à chaque noeud une fonction booléenne $f(u)$.
- Pour le noeud $u = 1$, $f(u) = 1$, pour le noeud 0 $f(u) = 0$ (valeurs booléennes).
- Pour tout noeud u, en supposant avoir associé $f(high(u))$ et $f(low(u))$, on aura :

$$f(u) = \big(val(u) \wedge f(high(u))\big) \vee \big(\neg var(u) \wedge f(low(u))\big)$$

- On dit alors que un BDD représente $f(racine)$.

Exemples (calcul inverse pour les deux BDDs construit antérieurement).

BDD réduit

Définition

Un BDD est réduit (ROBDD) si :

- Pour toute paire de noeuds non-terminaux u, v, si $var(u) = var(v)$, $low(u) = low(v)$ et $high(u) = high(v)$ alors $u = v$.
- Il n'existe pas de noeud inutile ou non-accessible en partant de la racine.

Les deux BDDs précédents sont réduits.

théorème

Pour toute fonction booléenne f et tout ordre sur les variables de f il existe un seul BDD réduit qui représente f.

Ordre des variables propositionnelles et taille de ROBDD

- L'ordre à partir duquel on construit l'arbre de décision binaire pour une formule a un impact très grand sur la taille du BDD réduit.
- Exemple : $(x_1 \wedge x_2) \vee (x_3 \wedge x_4)$, avec les deux ordres suivants :
  - $x_1 < x_2 < x_3 < x_4$.
  - $x_1 < x_3 < x_2 < x_4$.

  Construire les ROBDDs dans les deux cas, puis comparer.

Opérations sur les BDDs

- Supposons qu'on a construit un BDD réduit pour f (pour un ordre sur les variables de f).
- Comment calculer le BDD pour $\neg f$?
- Même question si on a des BDDs pour f_1 et f_2, et on veut un BDD pour $f_1 \wedge f_2$, $f_1 \vee f_2$, etc.
- Les résultats des opérations sont calculables inductivement sur la structure des BDDs.

Cas de la négation

- Il suffit de renverser 0 et 1 !
- Exemple, avec vérification...

Cas des deux opérations $\wedge$ et $\vee$

- Pour la conjonction, on utilise la propriété suivante :

$$\Big((x \wedge f_1) \vee (\neg x \wedge f_2)\Big) \wedge \Big((x \wedge f_3) \vee (\neg x \wedge f_4)\Big) = (x \wedge f_1 \wedge f_3) \vee (\neg x \wedge f_2 \wedge f_4)$$

- Pour la disjonction on aura une propriété similaire :

$$(x \wedge f_1) \vee (\neg x \wedge f_2) \vee (x \wedge f_3) \vee (\neg x \wedge f_4) = \Big(x \wedge (f_1 \vee f_3)\Big) \vee \Big(\neg x \wedge (f_2 \vee f_4)\Big)$$

- Ce qui veut dire que, pour trouver le BDD correspondant à $f \wedge g$, il suffit d'appliquer l'opération respective (mais avec qqs précautions) aux noeuds correspondants du BDD de f et de g.

Insertion d'un noeud dans un BDD

- Supposons qu'on veut insérer un noeud n dans un BDD donné D, en tant que père de deux noeuds $n_1 = \mathit{low}(n)$ et $n_2 = \mathit{high}(n)$, mais seulement si nécessaire.
- Le noeud à insérer est associé à une variable x.
- Les deux noeuds n_1 et n_2 doivent appartenir à D.
- Le noeud n'est pas nécessaire si $n_1 = n_2$!
  - On serait dans la situation d'un noeud inutile !
- Il ne serait non plus nécessaire s'il y a déjà un noeud n' dans D qui est père de n_1 et de n_2.
- Dans tous les autres cas, il faut l'insérer.
  - Car cela veut dire qu'aucun sous-BDD de D ne représente la formule $(\neg x \wedge f(u_1)) \vee (x \wedge f(u_2))$.
- Appelons cette procédure $\mathit{Ins}(n, x, D, n_1, n_2)$.

Algorithme pour $D_1 op D_2$, où $op \in \{\wedge, \vee\}$

- La procédure calcule, récursivement, un BDD correspondant à la conjonction (ou disjonction) $f_1 op f_2$, où f_1 correspond à un noeud u_1 dans D_1 et f_2 à u_2 dans D_2.
  - On l'appelle $Cons(op, u_1, u_2)$.
- On part avec $u_1 =$ la racine de D_1 et $u_2 =$ racine de D_2.
- Pour construire $Cons(op, u_1, u_2)$, quatre choix peuvent apparaître :
  1. u_1 et u_2 sont des noeuds terminaux. Alors on calcule $var(u_1)\, op\, var(u_2)$ et on renvoie le noeud respectiv (0 ou 1).
  2. $var(u_1) = var(u_2)$. Alors on construit, inductivement, $v_1 = Cons(op, high(u_1), high(u_2))$ et $v_2 = Cons(op, low(u_1), low(u_2))$ – tous les deux dans un même BDD D – et on renvoie $Ins((u_1, u_2), var(u_1), D, v_1, v_2)$.
  3. $var(u_1) < var(u_2)$. Alors on construit $v_1 = Cons(op, high(u_1), u_2)$ et $v_2 = Cons(op, low(u_1), u_2)$ dans un même BDD D et on renvoie $Ins((u_1, u_2), var(u_1), D, v_1, v_2)$.
  4. $var(u_1) > var(u_2)$. Alors on construit $v_1 = Cons(op, u_1, high(u_2))$ et $v_2 = Cons(op, u_1, low(u_2))$ – toujours dans un même BDD D – et on renvoie $Ins((u_1, u_2), var(u_2), D, v_1, v_2)$.

Exemples

Premier exemple : construire d'abord des ROBDDs pour chacune des formules suivantes et l'ordre $x_1 < x_2 < x_3 < x_4 < x_5$, puis appliquer l'algorithme *Cons* pour calculer le ROBDD pour $f_1 \wedge f_2$ avec le même ordre.

$$f_1 = ((\neg x_1 \wedge \neg x_2) \vee (x_1 \wedge x_2)) \wedge ((\neg x_3 \wedge \neg x_4) \vee (x_3 \wedge x_4)) \wedge \neg x_5$$
$$f_2 = ((\neg x_1 \wedge \neg x_3) \vee (x_1 \wedge x_3)) \wedge \neg x_5$$

Construire des BDDs pour les formules suivantes, d'abord en partant des arbres de décision binaire, puis en utilisant l'algorithme *Cons*, en considérant l'ordre $p_1 < p_2 < p_3$:

- $p_1 \vee p_2 \rightarrow p_1 \wedge p_2$.
- $\neg(p_1 \wedge p_2) \rightarrow (p_1 \vee p_3)$.