Révision des croyances dans la clôture propositionnelle de contraintes linéaires

Maela Laconi

Mai 2020

1 Introduction

Révision des croyances dans une clôture propositionnelle de contraintes linéaires.

2 20 avril

Une Constraint est définit par un double qui est son membre de droite ainsi que d'un Coefficients. Un coefficient est définit par une Variable et un Double, c'est une sous class de VariableValueTable.

Une interpreUne interpretation est une sous class de VariableValueTable. Elle contient la fonction satisfait, qui prend en paramètre une formule et qui fait appel a la fonction estSatisfaitPar qui prend en paramètre une interpretation et un objets Variables (recupéré à partir de formule). Variables est un ensemble de Variable.

Ce qui marche pour le moment :

- mise sous forme normale negative
- mise sous forme normale disjonctive
- interpretation (satisfait)

3 Fonctionnement toNNF

La fonction toNNF mets sous forme normale négative une formule. La négation peut être appliquée seulement à une contrainte.

3.1 Or

Dans le cas ou on applique toNNF a un or : on retourne un objet de type Or avec comme enfants les deux formules qui l'a compose auquelles on applique aussi toNNF.

3.2 And

Dans le cas ou on applique toNNF a un and :on retourne un objet de type And avec comme enfants les deux formules qui la compose auquelles on applique aussi toNNF.

3.3 Not

Dans le cas ou on applique toNNF a un not : on regarde si la formule qui le compose est une contrainte. Si la forme est une contrainte, alors on retourne juste cette contrainte (pour pas voir not(not(contraint) mais contrainte). Si c'est une formule binaire, alors on retourne la formule a laquelle on applique la fonction toSubNNF. Cette fonction toSubNNF est appelée lorsqu'on veut appliquer la transformation sous forme normale negative a une sous formule binaire (une formule qui est inclut dans une autre formule).

3.3.1 toSubNNF pour And

Dans le cas ou on applique toSubNNF a un and : on retoune un or de la negation de ses deux formules a laquelle on applique ensuite la fonction toNNF. On retourne un or car c'est la negation du and.

3.3.2 toSubNNF pour Or

Dans le cas ou on applique toSubNNF a un or : on retoune un and de la negation de ses deux formules a laquelle on applique ensuite la fonction toNNF. On retourne un and car c'est la negation du or.

3.4 Constraint

Dans le cas ou on applique toNNF a une contrainte, on retourne juste elle même (this).

4 Fonctionnement toDNF

La fonction toDNF mets sous forme normale disjonctive une formule. On transforme la formule en une disjonction de clauses conjonctives.

4.1 Or

Dans le cas ou on applique toDNF a un or : on retourne un or composé de son fils droit et gauche auquels on applique la transformation toDNF.

4.2 And

Dans le cas ou on applique toDNF a un and : soit a, b, c et d des formules.

4.2.1 Type a and b

On a a et b deux Constraint. Dans ce cas, on retourne juste elle même (this).

4.2.2 Type a and (b or c)

On a a une Constraint Dans ce cas, on retourne un or avec comme fils gauche un and composé du fils gauche (ici a) et du fils gauche du or (ici b) auquel on applique toDNF. Le fils droit du or est un and composé du fils gauche (ici a) et du fils droit du or (ici c) auquel on applique toDNF. On obtient donc : (a and b.toDNF()) or (a and c.toDNF()).

4.2.3 Type (a or b) and c

On a c une Constraint.

Dans ce cas, on retourne un or avec comme fils gauche un and composé du fils gauche du or (ici a) auquel on applique toDNF, et du fils droit (ici c). Le fils droit du or est un and composé du du fils droit du or (ici b) auquel on applique toDNF et du fils droit (ici c). On obtient donc : (a.toDNF() and c) or (b.toDNF() and c).

4.2.4 Type (a or b) and (c or d)

Dans ce cas, on construit 4 and. Un premier and1, composé du fils gauche du or de gauche (ici a) et pour fils droit le fils gauche du or a droite (ici c). On a and1 = (a and c). Un second and2, composé du fils gauche du or de gauche (ici a) et pour le fils droit le fils droit du or a droite (ici d). On a and2 = (a and d). Un troisième and3, composé du fils droit du or de gauche (ici b) et pour le fils droit le fils gauche du or de droite (ici c). On a and3 = (b and c). Un quatrième and4, composé du fils droit du or de gauche (ici b) et pour le fils droit le fils droit du or de droite (ici d). On a and4 = (b and d)

Ensuite, on un or avec comme fils gauche, un or composé de and1 et and2, auquels on applique toDNF. Et un second or composé de and3 et and4 auquels on applique également toDNF. On retourne donc ((and1.toDNF() or and2.toDNF()) or (and3.toDNF() or and4.toDNF()))

4.2.5 Le cas par défaut

Par défaut, on retoune un and composé du fils gauche et droit auquels on applique toDNF (cas ou on a un and composé d'autres and).

4.3 Not

Dans le cas ou on applique toDNF a un not : on retourne juste elle même (this).

if $\psi.estIncoherent()||\mu.estIncoherent()$ **then**
  $\varrho = \mu$
  $d^* = +\infty$
end if
$\widehat{\psi} = \psi.toConjonction()$
$\widehat{\mu} = \mu.toConjonction()$
$d^* = \widehat{\psi} *^{d_\varepsilon} \widehat{\mu}$
if $d^* \in \varepsilon\mathbf{N}$ **then**
  $\lambda = d^*$
else
  $\lambda = \varepsilon \left\lceil \frac{d^*}{\varepsilon} \right\rceil$
end if
return ϱ

Figure 1: $*^{d_\varepsilon}$

4.4 Constraint

Dans le cas ou on applique toDNF a une contrainte, on retourne juste elle même (this).

5 algorithme de $*^{d\varepsilon}$ dans L_{CPCL}

Soit $\psi, \mu \in L_{CPCL}$. Le calcul de $\varrho = \psi *^{d\varepsilon} \mu$ est décrit pour trois cas, chacun des deux derniers cas faisant appel au précédent.

5.1 Cas ou ψ et μ sont des conjonctions de littéraux

Si ψ ou μ est incohérent alors $\psi *^{d\varepsilon} \mu \equiv \mu$ avec $d = +\infty$.

5.2 Cas ou ψ et μ sont sous forme normale négative

Soit ψ et μ deux formules sous forme normale disjonctive. ψ est donc de la forme $\bigvee \psi_k$ et $\mu = \bigvee \mu_l$. Soit $d*_{kl} = d(\psi, \mu)$

6 Algorithmes

6.1 Algorithme pour le calcul de la distance de Manhattan

6.2 Algorithme pour le calcul de $d_\varepsilon(\psi, \mu)$

int d
for int j = 1 ; j ¡= n ; j++ **do**
  d = d + $w_j|\mu_j - \psi_j|$
end for
return d

Figure 2: Distance de Manhattan d(ψ, μ)

return $\varepsilon \left\lceil \frac{d(\psi,\mu}{\varepsilon} \right\rceil$

Figure 3: $d_\varepsilon(\psi, \mu)$

6.3 Algorithme pour le calcul de $d^*(\psi, \mu)$

6.4 Algorithme de transformation large (remplacement des iné- galités strictes par des inégalités larges ou élimination du not sur une contrainte)

6.5 Algorithme qui effectue la negation sur les coefficients d'une contrainte (dans la classe Constraint)

6.6 Algorithme qui effectue la negation sur les coefficients d'une contrainte (dans la classe Coefficient)

6.7 Algorithme qui effectue l'opposé d'un RationalNumber (dans la classe RationalNumber)

6.8 Algorithme $*^{d_\varepsilon}(\varepsilon, \psi, \mu$ dans le cas où $\psi et \mu$ sont des conjonctions de littéraux

$\widehat{\psi} = \psi.large()$
$\widehat{\mu} = \mu.large()$
$(\varrho, d) = d(\widehat{\psi}, \widehat{\mu})$
return d

Figure 4: $d^*(\psi, \mu)$

if $\psi.isNegativeConstraint()$ **then**
  return $\psi.toOppositeCoefficients()$
end if
if $\psi.isConstraint()$ **then**
  return ψ
end if
if $\psi.isNot()$ **then**
  $\psi.setChild(large(\psi.getChild))$
  return ψ
end if
if $\psi.isBinaryFormula()$ **then**
  $\psi.setLeftChild(large(\psi.getLeftChild))$
  $\psi.setRightChild(large(\psi.getRightChild))$
  return ψ
end if
return ψ

Figure 5: $large(\psi)$

return this.coefficient.toOppositeCoefficients()

Figure 6: toOppositeCoefficients()

for (Map.Entry me : tab.entrySet() **do**
  me.getValue() = me.getValue().toOpposite()
end for

Figure 7: toOppositeCoefficients()

this.value = this.value * (-1)

Figure 8: toOpposite()

if $\psi.estIncoherent() || \mu.estIncoherent()$ **then**
  $\varrho = \mu$
  $d^*_\varepsilon = +\infty$
  return $(\varrho, d^*_\varepsilon)$
end if
$d^* = d^*(\psi, \mu)$
calcul de ψ'
if $d^* \in \varepsilon\mathbf{N}$ **then**
  $\lambda_\varepsilon = d^*$
  if $(\psi' \bigwedge \mu).isCoherent()$ **then**
    $d^*_\varepsilon = d^*$
    $\varrho = \psi' \bigwedge \mu$
  else
    $\lambda_\varepsilon = d^* + \varepsilon$
    calcul de ψ'
    $\varrho = \psi' \bigwedge \mu$
  end if

else
  $\lambda_\varepsilon = \varepsilon \left\lceil \frac{d^*}{\varepsilon} \right\rceil$ $\varrho = \psi' \bigwedge \mu$ $d^*_\varepsilon = \lambda_\varepsilon$
end if
return $(\varrho, d^*_\varepsilon)$

Figure 9: $*^{d_\varepsilon}(\varepsilon, \psi, \mu$

$\varrho = \bigvee_{d_\varepsilon^*} \psi_k *^{d_\varepsilon} \mu_l$
$d_\varepsilon^* = d_\varepsilon(\psi, \mu)$
return $(\varrho, d_\varepsilon^*)$

Figure 10: $*^{d_\varepsilon}(\varepsilon, \psi, \mu$

return $*^{d_\varepsilon}(\varepsilon, \psi.toDNF(), \mu.toDNF()$

Figure 11: $*^{d_\varepsilon}(\varepsilon, \psi, \mu$

6.9 Algorithme $*^{d_\varepsilon}(\varepsilon, \psi, \mu$ dans le cas où $\psi et \mu$ sont sous DNF

6.10 Algorithme $*^{d_\varepsilon(\varepsilon, \psi, \mu)}$ dans le cas général

6.11 Calcul de ψ'

Le calcul de ψ' se fait étant donné un rationnel $\lambda \geq 0$ et une conjonction de littéraux ψ, et retourne une formule ψ_k avec $M(\psi_k) = G_\lambda^d(M(\psi))$. $G_\lambda^d(M(\psi))$ étant la dilatation de $M(\psi)$ à distance λ pour la distance d.

On a :

$\phi = x \in M(\psi) \wedge (z_j \geq y_j - x_j) \wedge (z_j \geq x_j - y_j) \wedge (w_1 z_1 \leq \lambda + ... + w_n z_n \leq \lambda)$

L'algorithme du calcul de ψ' prend $\lambda \leq 0$ et une conjonction de littéraux, ici ϕ ci-dessus, en entrée. Il retourne ϕ_k telle que $M(\phi_k) = G_\lambda^d(M(\phi))$.

$x \in M(\psi)$:

- Soit ψ une contrainte linéaire, si $\psi = u_1 + u_2 \leq 5$ et que $x \in M(\psi)$ alors on obtient $x_1 + x_2 \leq 5$;
- Soit ψ une contrainte linéaire stricte, si $\psi = u_1 + u_2 \leq 4$, ce qui revient à $\neg(-u_1 - u_2 \leq -4)$ et que $x \in M(\psi)$ alors on obtient $\neg(-x_1 - x_2 \leq -4)$;
- Soit ψ une conjonction de littéraux, si $\psi = u_1 + u_2 \leq 10 \wedge u_1 - u_2 \leq 5$, ce qui revient à $\psi = u_1 + u_2 \leq 10 \wedge \neg(-u_1 + u_2 \leq -5)$, et que $x \in M(\psi)$ alors on obtient $x_1 + x_2 \leq 10 \wedge \neg(-x_1 + x_2 \leq -5)$.

6.12 EstCoherent

Si ψ est coherent alors il existe une interpretations qui le safisait. C'est-à-dire que $M(\psi) \neq \theta$

Soit I une interprétation et ψ une formule. Le fait que I satisfasse ψ peut être défini intuitivement par le fait de remplacer chaque variable x par la valeur x^I, chaque connecteur par un opérateur sur les booléens ($\neg$ par non, $\wedge$ par et, $\vee$ par ou, etc.) et à évaluer l'expression.

7 Diagramme de classes