Raisonner avec égalité

Simon Robillard

2022

Section 1

Congruence closure

Introduction

Lors des semaines précédentes, vous avez vu comment un solveur SMT fait appel à différent solveurs pour différentes théories

- théorie de l'arithmétique $\rightarrow$ algorithme du simplexe (pas détaillé dans ce cours)
- théorie de l'égalité sur des fonctions non-interprétées $\rightarrow$ algorithme *congruence closure*
- ...

Notation

- dans ces slides, $\approx$ dénote le **prédicat** d'égalité
  - pour le distinguer de l'égalité "méta-théorique" dénotée par $=$
  - notation infixe : $s \approx t$ plutôt que $\approx(s, t)$
  - $\neg s \approx t$ est aussi noté $s \not\approx t$
- $\mathcal{T}(\Sigma, \mathcal{V})$ dénote l'ensemble des termes du premier ordre formés à partir de
  - la signature Σ (l'ensemble de symboles de prédicats, de constantes et de fonctions)
  - l'ensemble de variables $\mathcal{V}$

Le problème

- on considère une conjonction d'égalité et d'inégalités

$$\bigwedge_{i=0}^{n} s_i \approx t_i \wedge \bigwedge_{j=0}^{m} s_j \not\approx t_j$$

- on cherche à déterminer si cette conjonction est satisfiable
  - contrairement au problème d'unification, il ne s'agit pas de déterminer si les égalités respectent l'égalité syntactique

    $$f(a) \approx g(b) \text{ est satisfiable}$$

  - il s'agit uniquement de vérifier que la propriété des fonctions est respectée

    $$a \approx b \wedge f(a) \not\approx f(b) \text{ n'est pas satisfiable}$$

  - les variables $x \in \mathcal{V}$ sont interprétées existentiellement

Quelques exemples

1. $a \approx b \wedge a \approx c \wedge f(a, c) \not\approx f(b, a)$

2. $x \approx a \wedge x \not\approx f(a)$

3. $a \approx f(c) \wedge b \approx c \wedge g(a, x) \not\approx g(f(b), x)$

4. $a \approx b \wedge b \approx c \wedge g(f(a), b) \approx g(f(c), a) \wedge f(a) \not\approx b$

Quelques exemples

1. $a \approx b \land a \approx c \land f(a,c) \not\approx f(b,a)$
   unsat
2. $x \approx a \land x \not\approx f(a)$
3. $a \approx f(c) \land b \approx c \land g(a,x) \not\approx g(f(b),x)$
4. $a \approx b \land b \approx c \land g(f(a),b) \approx g(f(c),a) \land f(a) \not\approx b$

Quelques exemples

1. $a \approx b \land a \approx c \land f(a, c) \not\approx f(b, a)$
   unsat
2. $x \approx a \land x \not\approx f(a)$
   sat (exemple d'interprétation avec deux éléments)
3. $a \approx f(c) \land b \approx c \land g(a, x) \not\approx g(f(b), x)$

4. $a \approx b \land b \approx c \land g(f(a), b) \approx g(f(c), a) \land f(a) \not\approx b$

Quelques exemples

1. $a \approx b \wedge a \approx c \wedge f(a,c) \not\approx f(b,a)$
   unsat
2. $x \approx a \wedge x \not\approx f(a)$
   sat (exemple d'interprétation avec deux éléments)
3. $a \approx f(c) \wedge b \approx c \wedge g(a,x) \not\approx g(f(b),x)$
   unsat
4. $a \approx b \wedge b \approx c \wedge g(f(a),b) \approx g(f(c),a) \wedge f(a) \not\approx b$

Quelques exemples

1. $a \approx b \wedge a \approx c \wedge f(a, c) \not\approx f(b, a)$
   unsat
2. $x \approx a \wedge x \not\approx f(a)$
   sat (exemple d'interprétation avec deux éléments)
3. $a \approx f(c) \wedge b \approx c \wedge g(a, x) \not\approx g(f(b), x)$
   unsat
4. $a \approx b \wedge b \approx c \wedge g(f(a), b) \approx g(f(c), a) \wedge f(a) \not\approx b$
   sat (exemple d'interprétation avec trois éléments)

Quelques exemples

1. $a \approx b \land a \approx c \land f(a, c) \not\approx f(b, a)$
   unsat
2. $x \approx a \land x \not\approx f(a)$
   sat (exemple d'interprétation avec deux éléments)
3. $a \approx f(c) \land b \approx c \land g(a, x) \not\approx g(f(b), x)$
   unsat
4. $a \approx b \land b \approx c \land g(f(a), b) \approx g(f(c), a) \land f(a) \not\approx b$
   sat (exemple d'interprétation avec trois éléments)

Observation : les variables sont quantifiées existentiellement. En pratique, pour ce problème, on les traite exactement comme des constantes.

Le principe de *congruence closure*

- on considère uniquement les termes qui apparaissent dans le problème (les s_i, t_i, ... mais aussi leurs sous-termes propres)
- on cherche une partition maximale de ces termes en classes d'équivalence, tout en respectant
  - les égalités : si le problème contient $s \approx t$, alors s et t doivent appartenir à une même classe C
  - la congruence : si pour tout i t.q. $1 \leq i \leq n$, s_i et t_i appartiennent chacun à une classe C_i, alors $f(s_1, \ldots, s_n)$ et $f(t_1, \ldots, t_n)$ doivent appartenir à une même classe C
- le problème est insatisfiable si et seulement si il contient une inégalité $s \not\approx t$ telle que s et t appartiennent à la même classe

Algorithme CC

1. pour chaque terme t

   $\text{class}(t) := \{t\}$
2. pour chaque égalité $s_i \approx t_i$

   $\text{class}(s_i) := \text{class}(t_i) := \text{class}(s_i) \cup \text{class}(t_i)$
3. tant qu'il existe $f(s_1, \ldots, s_n)$ et $f(t_1, \ldots, t_n)$ t.q.
   - $\text{class}(f(\bar{s})) \neq \text{class}(f(\bar{t}))$ et
   - pour tout i t.q. $0 \leq i \leq n$, $\text{class}(s_i) = \text{class}(t_i)$

   $\text{class}(f(\bar{s})) := \text{class}(f(\bar{t})) := \text{class}(f(\bar{s})) \cup \text{class}(f(\bar{t}))$
4. si il existe une inégalité $s_i \not\approx t_i$ t.q. $\text{class}(s_i) = \text{class}(t_i)$

   retourner unsat
5. sinon

   retourner sat

Exemple 1

Problème : $a \approx b \wedge a \approx c \wedge f(a, c) \not\approx f(b, a)$

initialisation	$\{a\},\ \{b\},\ \{c\},\ \{f(a, c)\},\ \{f(b, a)\}$

Exemple 1

Problème : $a \approx b \wedge a \approx c \wedge f(a, c) \not\approx f(b, a)$

initialisation	$\{a\}$, $\{b\}$, $\{c\}$, $\{f(a, c)\}$, $\{f(b, a)\}$
$a \approx b$	$\{a, b\}$, $\{c\}$, $\{f(a, c)\}$, $\{f(b, a)\}$

Exemple 1

Problème : $a \approx b \wedge a \approx c \wedge f(a,c) \not\approx f(b,a)$

initialisation	$\{a\},\ \{b\},\ \{c\},\ \{f(a,c)\},\ \{f(b,a)\}$
$a \approx b$	$\{a,b\},\ \{c\},\ \{f(a,c)\},\ \{f(b,a)\}$
$a \approx c$	$\{a,b,c\},\ \{f(a,c)\},\ \{f(b,a)\}$

Exemple 1

Problème : $a \approx b \wedge a \approx c \wedge f(a, c) \not\approx f(b, a)$

initialisation	$\{a\},\ \{b\},\ \{c\},\ \{f(a,c)\},\ \{f(b,a)\}$
$a \approx b$	$\{a,b\},\ \{c\},\ \{f(a,c)\},\ \{f(b,a)\}$
$a \approx c$	$\{a,b,c\},\ \{f(a,c)\},\ \{f(b,a)\}$
congruence $f(a,c)$ et $f(b,a)$	$\{a,b,c\},\ \{f(a,c), f(b,a)\}$

Résultat : **unsat** car $\text{class}(f(a,c)) = \text{class}(f(b,a))$

Exemple 2

Problème : $a \approx b \wedge b \approx c \wedge g(f(a), b) \approx g(f(c), a) \wedge f(a) \not\approx b$

initialisation	$\{a\}$, $\{b\}$, $\{c\}$, $\{f(a)\}$, $\{f(c)\}$, $\{g(f(a), b)\}$, $\{g(f(c), a)\}$

Exemple 2

Problème : $a \approx b \land b \approx c \land g(f(a), b) \approx g(f(c), a) \land f(a) \not\approx b$

initialisation	$\{a\}$, $\{b\}$, $\{c\}$, $\{f(a)\}$, $\{f(c)\}$, $\{g(f(a), b)\}$, $\{g(f(c), a)\}$
$a \approx b$	$\{a, b\}$, $\{c\}$, $\{f(a)\}$, $\{f(c)\}$, $\{g(f(a), b)\}$ $\{g(f(c), a)\}$

Exemple 2

Problème : $a \approx b \wedge b \approx c \wedge g(f(a), b) \approx g(f(c), a) \wedge f(a) \not\approx b$

initialisation	$\{a\}, \{b\}, \{c\}, \{f(a)\}, \{f(c)\}, \{g(f(a), b)\}, \{g(f(c), a)\}$
$a \approx b$	$\{a, b\}, \{c\}, \{f(a)\}, \{f(c)\}, \{g(f(a), b)\}\ \{g(f(c), a)\}$
$b \approx c$	$\{a, b, c\}, \{f(a)\}, \{f(c)\}, \{g(f(a), b)\}\ \{g(f(c), a)\}$

Exemple 2

Problème : $a \approx b \wedge b \approx c \wedge g(f(a), b) \approx g(f(c), a) \wedge f(a) \not\approx b$

initialisation	$\{a\}, \{b\}, \{c\}, \{f(a)\}, \{f(c)\}, \{g(f(a), b)\}, \{g(f(c), a)\}$
$a \approx b$	$\{a, b\}, \{c\}, \{f(a)\}, \{f(c)\}, \{g(f(a), b)\}\ \{g(f(c), a)\}$
$b \approx c$	$\{a, b, c\}, \{f(a)\}, \{f(c)\}, \{g(f(a), b)\}\ \{g(f(c), a)\}$
$g(f(a), b) \approx g(f(c), a)$	$\{a, b, c\}, \{f(a)\}, \{f(c)\}, \{g(f(a), b), g(f(c), a)\}$

Exemple 2

Problème : $a \approx b \land b \approx c \land g(f(a), b) \approx g(f(c), a) \land f(a) \not\approx b$

initialisation	$\{a\},\ \{b\},\ \{c\},\ \{f(a)\},\ \{f(c)\},\ \{g(f(a), b)\},\ \{g(f(c), a)\}$
$a \approx b$	$\{a, b\},\ \{c\},\ \{f(a)\},\ \{f(c)\},\ \{g(f(a), b)\}\ \{g(f(c), a)\}$
$b \approx c$	$\{a, b, c\},\ \{f(a)\},\ \{f(c)\},\ \{g(f(a), b)\}\ \{g(f(c), a)\}$
$g(f(a), b) \approx g(f(c), a)$	$\{a, b, c\},\ \{f(a)\},\ \{f(c)\},\ \{g(f(a), b), g(f(c), a)\}$
congruence $f(a)$ et $f(c)$	$\{a, b, c\},\ \{f(a), f(c)\},\ \{g(f(a), b), g(f(c), a)\}$

Résultat : **sat** car $\text{class}(f(a)) \neq \text{class}(b)$

Implémentation efficace

- Principe de base proposé par Shostak (1977)
- Nelson et Oppen proposent la même année une implémentation efficace :
  - utiliser union-find pour représenter les classes d'équivalence
  - représenter l'ensemble des termes sous forme de graphe pour trouver efficacement les congruences à propager
  - complexité quadratique

Section 2

Superposition

Introduction

- Nous avons vu comment raisonner sur l'égalité sur des formules existentiellement quantifiées
- Comment faire pour raisonner sur l'égalité en logique du premier ordre (avec $\forall$ et $\exists$) ?
  - il est possible d'ajouter de nouvelles règles à la méthode des tableaux (pas couvert ici)
  - il est possible de faire de la saturation avec des règles d'inférences dédiées à l'égalité (c'est l'objet de ce cours)

L'égalité pour seul prédicat

On fait l'hypothèse que $\approx$ est le seul prédicat dans la signature

- pas essentiel, mais permet de limiter le nombre de règles d'inférences
- ne limite pas l'expressivité de la logique du premier ordre !

Encodage des prédicats

1. remplacer chaque symbole de prédicat p par un symbole de fonction f_p de même arité
2. ajouter à la signature une constante $\top$
3. remplacer chaque atome $p(t_1, \dots, t_n)$ par l'atome $f_p(t_1, \dots, t_n) \approx \top$

Rappel : axiomatisation de la théorie de l'égalité

- réflexivité

$$x \approx x$$

- symétrie

$$x \not\approx y \vee y \approx x$$

- transitivité

$$x \not\approx y \vee y \not\approx z \vee x \approx z$$

- monotonicité (pour chaque symbole de fonction f)

$$x_1 \not\approx y_1 \vee \cdots \vee x_n \not\approx y_n \vee f(x_1, \ldots, x_n) \approx f(y_1, \ldots, y_n)$$

- monotonicité (pour chaque symbole de prédicat p)

$$x_1 \not\approx y_1 \vee \cdots \vee x_n \not\approx y_n \vee \not p(x_1, \ldots, x_n) \vee p(y_1, \ldots, y_n))$$

Une théorie très explosive

- il est possible de raisonner *modulo égalité* en ajoutant les axiomes aux hypothèses du problème
- saturer cet ensemble de clauses génère un très grand nombre de nouvelles conclusions
- très peu de ces conclusions sont utiles pour parvenir à la réfutation

Paramodulation

$$\frac{s \approx t \vee C_1 \qquad C_2[s']}{\sigma(C_1 \vee C_2[t])}$$

- σ est un *most general unifier (mgu)* de s et s'
- dans cette règle (et dans la suite de la présentation), on néglige l'ordre des arguments du prédicat $\approx$, càd que $s \approx t$ et $t \approx s$ dénotent le même litéral

Résolution d'égalité

$$\frac{s \not\approx s' \vee C}{\sigma(C)}$$

- σ est un *most general unifier (mgu)* de s et s'
- cette règle permet de résoudre les contradictions

Paramodulation + ordres de terme = superposition

- avec la paramodulation, les égalités peuvent être appliquées dans les deux sens
- une technique standard pour raisonner efficacement avec l'égalité est d'orienter les équations et de les traiter comme des règles de réécriture
  - il faut s'assurer que les réécritures terminent
  - c'est le cas si chaque règle de réécriture $s \to t$ vérifie $s \succ t$, où $\succ$ est un ordre **bien fondé**

Exemple

- pour le système de réécriture $\{a \to b, b \to c, c \to a\}$, il n'existe pas d'ordre $\succ$ bien fondé $\implies$ le système ne termine pas
- pour le système de réécriture $\{a \to b, b \to c, a \to c\}$, on a l'ordre bien fondé $a \succ b \succ c$, le système termine

Ordres de termes

Propriété

1. $\succ$ est bien fondé (il n'existe pas de chaîne infinie $t \succ t' \succ t'' \succ \dots$)
2. $\succ$ est compatible avec l'instantiation : pour tous termes s et t et toute substitution σ, si $s \succ t$ alors $\sigma(s) \succ sigma(t)$
3. $\succ$ est total

Ordres de termes

Propriété

1. $\succ$ est bien fondé (il n'existe pas de chaîne infinie $t \succ t' \succ t'' \succ \dots$)
2. $\succ$ est compatible avec l'instantiation : pour tous termes s et t et toute substitution σ, si $s \succ t$ alors $\sigma(s) \succ sigma(t)$
3. $\succ$ est total sur les termes sans variables

Pas d'ordre total sur tous les termes avec variable

- ordonner y et $f(x)$?
- pour être bien fondé, $\succ$ doit respecter la propriété du sous-terme : pour tout t, $f(t) \succ t$
- impossible d'être compatible avec $\sigma = \{x \mapsto y\}$ et $\sigma' = \{y \mapsto f(f(x))\}$

Ordres de termes et superposition

- on fixe un ordre sur les termes, avec les bonnes propriétés
  - exemples : Knuth-Bendix Ordering (KBO), Lexicographic Path Ordering (LPO), Multiset Path Ordering (LPO), ...
- la règle de superposition est similaire à la paramodulation, mais applique la réécriture uniquement si elle ne crée pas un terme “plus grand”
  - puisque l’ordre sur les termes n’est pas total, on ne peut pas imposer la condition $s \succ t$
  - à la place, on utilise une condition moins stricte : $s \not\preceq t$

Les règles de superposition

Superposition positive et négative

$$\frac{s \approx t \vee C_1 \qquad u[s'] \approx v \vee C_2}{\sigma(u[t] \approx v \vee C_1 \vee C_2)}\ \mathsf{Sup}^+$$

$$\frac{s \approx t \vee C_1 \qquad u[s'] \not\approx v \vee C_2}{\sigma(u[t] \not\approx v \vee C_1 \vee C_2)}\ \mathsf{Sup}^-$$

- σ is an mgu of s and s'
- $\sigma(s) \not\preceq \sigma(t)$
- $\sigma(u[s']) \not\preceq \sigma(v)$
- s' is not a variable

Résolution d'égalité

$$\frac{s \not\approx s' \vee C}{\sigma(C)}\ \mathsf{EqRes}$$

- σ is an mgu of s and s'

Note : pour être réfutationellement complet, le calcul de superposition a besoin d'une règle supplémentaire appelée **factorisation d'égalité**. Cette règle sert assez rarement en pratique, elle est omise ici par souci de simplification.

Exemple 3

Supposons l'ordre suivant sur les termes : $g(a, b) \succ d \succ c \succ b \succ a$

$$\frac{b \approx a \qquad g(a, b) \not\approx b \vee d \approx c}{?}$$

Exemple 3

Supposons l'ordre suivant sur les termes : $g(a,b) \succ d \succ c \succ b \succ a$

$$\frac{b \approx a \qquad g(a,b) \not\approx b \vee d \approx c}{?}$$

- la superposition produit la clause $g(a,a) \not\approx b \vee d \approx c$
- certaines inférences possibles avec la paramodulation sont bloquées par les conditions d'ordre de la superposition
  - $g(b,b) \not\approx b \vee d \approx c$
    la superposition n'utilise l'égalité $b \approx a$ que dans un sens
    (l'autre sens n'est pas applicable car $b \succ a$)
  - $g(a,b) \not\approx a \vee d \approx c$
    la superposition ne réécrit pas le "petit côté" des (in)égalités

Exemple 4

Supposons l'ordre suivant sur les termes sans variables : $c \succ b \succ a$
Deux termes s et t sont incomparables si $\mathrm{var}(s) \neq \mathrm{var}(t)$

$$\frac{g(x, x) \approx a \qquad g(b, y) \not\approx b \vee y \not\approx c}{?}$$

Exemple 4

Supposons l'ordre suivant sur les termes sans variables : $c \succ b \succ a$
Deux termes s et t sont incomparables si $\text{var}(s) \neq \text{var}(t)$

$$\frac{g(x,x) \approx a \qquad g(b,y) \not\approx b \vee y \not\approx c}{?}$$

- conditions d'ordre
  - $g(x,x)$ n'est pas comparable à a, il faut donc tester les réécritures possibles dans les deux sens
  - les deux côtés des deux inégalités sont incomparables, ils peuvent donc tous deux être la cible de réécritures
- la superposition produit la clause $a \not\approx b \vee b \not\approx c$
  - $\sigma(a \not\approx b \vee y \not\approx c)$ avec $\sigma = \{x \mapsto b, y \mapsto b\}$
- la superposition ne réécrit pas les variables
  - sans cette restriction on pourrait unifier y avec a ou avec $g(x,x)$ et produire plusieurs autres conclusions

Algorithme de saturation

- l'algorithme de saturation est le même que pour la résolution
  - pour que la recherche de preuve soit (réfutationellement) complète, la saturation doit être équitable : aucune inférence possible ne doit être reportée indéfiniment
  - pour garantir cela, il faut choisir comme *given clause* la plus ancienne des clauses (on gère l'ensemble des clauses à traiter comme une file)
- avec l'égalité, la détection des tautologies n'est pas triviale
  - $a \approx a$
  - $a \not\approx b \vee f(a) \approx f(b)$
  - $a \not\approx b \vee a \not\approx c \vee b \approx c$
- on utilise *congruence closure* pour détecter les tautologies

Factorisation d'égalité

$$\frac{s \approx t \vee s' \approx u \vee C}{\sigma(s \approx t \vee t \not\approx u \vee C)}\ \text{EqFact}$$

- σ est un *most general unifier (mgu)* de s et s'
- $\sigma(s) \not\preceq \sigma(t)$ et $\sigma(t) \not\preceq \sigma(u)$
- intuition de la conclusion : si t et u sont égaux sémantiquement et s et s' sont égaux syntactiquement (σ), alors on peut "oublier" une des deux égalités de la prémisse
- bien que rarement utilisée, cette règle est nécessaire pour la complétude réfutationnelle du calcul