Logică și structuri discrete

Logica predicatelor. Algoritm de unificare: union-find

Casandra Holotescu

casandra@cs.upt.ro

https://tinyurl.com/lecturesLSD

Cum se face unificarea?

În logica predicatelor, rezoluția *unifică* un literal cu negatul lui:

$$A \vee P(t_1, t_2, ..., t_n) \quad \text{și} \quad B \vee \neg P(t'_1, t'_2, ..., t'_n)$$

dacă putem unifica ("potrivi") argumentele lui P și $\neg P$: t_1 cu t'_1,...

Cum se face unificarea?

În logica predicatelor, rezoluția *unifică* un literal cu negatul lui:

$$A \vee P(t_1, t_2, ..., t_n) \quad \text{și} \quad B \vee \neg P(t'_1, t'_2, ..., t'_n)$$

dacă putem unifica ("potrivi") argumentele lui P și $\neg P$: t_1 cu t'_1,...

O *substituție* e o *funcție* care asociază unor *variabile* niște *termeni*:

$$\{x_1 \mapsto t_1, \ \ldots, x_n \mapsto t_n\}$$

Doi termeni se pot *unifica* dacă există o substituție care îi face egali (o asemenea substituție se numește *unificator*)

$$f(x, g(y, z), t)\{x \mapsto h(z), y \mapsto h(b), t \mapsto u\} = f(h(z), g(h(b), z), u)$$

Cum se face unificarea?

În logica predicatelor, rezoluția *unifică* un literal cu negatul lui:

$$A \vee P(t_1, t_2, ..., t_n) \quad \text{și} \quad B \vee \neg P(t'_1, t'_2, ..., t'_n)$$

dacă putem unifica ("potrivi") argumentele lui P și $\neg P$: t_1 cu t'_1,...

O *substituție* e o *funcție* care asociază unor *variabile* niște *termeni*:

$$\{x_1 \mapsto t_1, \ \ldots, x_n \mapsto t_n\}$$

Doi termeni se pot *unifica* dacă există o substituție care îi face egali (o asemenea substituție se numește *unificator*)

$$f(x, g(y, z), t)\{x \mapsto h(z), y \mapsto h(b), t \mapsto u\} = f(h(z), g(h(b), z), u)$$

Termenul T cu substituția σ se notează uzual postfix: $T\sigma$

Substituția găsită se aplică predicatelor care rămân (rezolventul):

$$\frac{A \vee P(t_1, t_2, ..., t_n) \qquad B \vee \neg P(t'_1, t'_2, ..., t'_n)}{A\sigma \vee B\sigma}$$

Reguli de unificare

O *variabilă* x poate fi unificată cu orice *termen* t (substituție)
dacă x *nu apare* în t (altfel, substituind obținem un termen infinit)
deci nu: x cu $f(h(y), g(x, z))$; dar trivial, putem unifica x cu x

Doi *termeni* $f(...)$ pot fi unificați doar dacă au aceeași funcție,
și *argumentele* (termeni) pot fi unificate (poziție cu poziție)

Două *constante* (funcții cu 0 arg.) $\Rightarrow$ unificate dacă sunt identice

$\Rightarrow$ cu aceste reguli, putem găsi *cel mai general unificator*
(orice alt unificator se poate obține din el printr-o altă substituție)

Clase de echivalență și Union-Find

Dacă unificăm pe x cu y și apoi pe y cu $f(z, a)$,
atunci si x e unificat cu $f(z, a) \Rightarrow$ *relație de echivalență*

Clase de echivalență și Union-Find

Dacă unificăm pe x cu y și apoi pe y cu $f(z, a)$,
atunci si x e unificat cu $f(z, a) \Rightarrow$ *relație de echivalență*

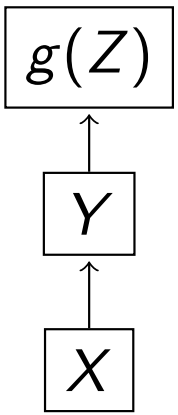
Trebuie să reținem ce variabile sunt *echivalente*.

Union-Find:
structură de date+algoritm pentru lucru cu clase de echivalență.

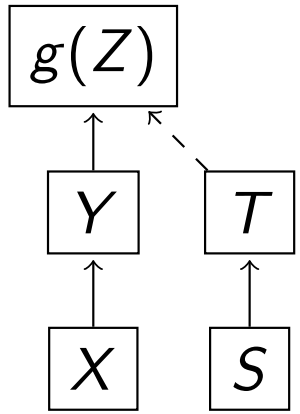
Operații:
find (element): găsește reprezentantul clasei de echivalență
union (elem1, elem2): declară elementele ca fiind echivalente
(rămân echivalente mai departe)

Exemplu pentru Union-Find

O implementare: pădure de *arbori* cu legături de la fiu la părinte
find: returnează rădăcina (chiar nodul, dacă e singur)
union: leagă rădăcina unuia de rădăcina celuilalt



$find(X) = find(Y) = g(Z)$



$union(Y, S)$
leagă $find(Y)$ și $find(S)$

Union-Find cu dicționare

Substituție = *dicționar* de la variabile (string) la termeni

union(D, T_1, T_2) produce *o nouă substituție* care unifică T_1 cu T_2
pornind de la o substituția / dicționarul D existent
construim substituția pas cu pas, unificând succesiv termeni

find(D, X) caută recursiv reprezentantul lui X ("ce înseamnă")
adică aplică lui X substituțiile din D

Union-Find cu dicționare

Substituție = *dicționar* de la variabile (string) la termeni

union(D, T_1, T_2) produce *o nouă substituție* care unifică T_1 cu T_2
pornind de la o substituția / dicționarul D existent
construim substituția pas cu pas, unificând succesiv termeni

find(D, X) caută recursiv reprezentantul lui X ("ce înseamnă")
adică aplică lui X substituțiile din D

Exemplu: unificăm $f(x, g(x, s(z)), t)$ cu $f(h(z), g(h(b), u), z)$

Union-Find cu dicționare

Substituție = *dicționar* de la variabile (string) la termeni

union(D, T_1, T_2) produce *o nouă substituție* care unifică T_1 cu T_2
pornind de la o substituția / dicționarul D existent
construim substituția pas cu pas, unificând succesiv termeni

find(D, X) caută recursiv reprezentantul lui X ("ce înseamnă")
adică aplică lui X substituțiile din D

Exemplu: unificăm $f(x, g(x, s(z)), t)$ cu $f(h(z), g(h(b), u), z)$
x cu $h(z) \Rightarrow \{x \mapsto h(z)\}$

Union-Find cu dicționare

Substituție = *dicționar* de la variabile (string) la termeni

union(D, T_1, T_2) produce *o nouă substituție* care unifică T_1 cu T_2
pornind de la o substituția / dicționarul D existent
construim substituția pas cu pas, unificând succesiv termeni

find(D, X) caută recursiv reprezentantul lui X ("ce înseamnă")
adică aplică lui X substituțiile din D

Exemplu: unificăm $f(x, g(x, s(z)), t)$ cu $f(h(z), g(h(b), u), z)$
x cu $h(z) \Rightarrow \{x \mapsto h(z)\}$
$g(x, s(z))$ cu $g(h(b), u) \Rightarrow$
x cu $h(b) \Rightarrow h(z)$ cu $h(b) \Rightarrow \{x \mapsto h(z), z \mapsto b\}$
$s(z)$ cu $u \Rightarrow \{x \mapsto h(z), z \mapsto b, u \mapsto s(z)\}$

Union-Find cu dicționare

Substituție = *dicționar* de la variabile (string) la termeni

union(D, T_1, T_2) produce *o nouă substituție* care unifică T_1 cu T_2
pornind de la o substituția / dicționarul D existent
construim substituția pas cu pas, unificând succesiv termeni

find(D, X) caută recursiv reprezentantul lui X ("ce înseamnă")
adică aplică lui X substituțiile din D

Exemplu: unificăm $f(x, g(x, s(z)), t)$ cu $f(h(z), g(h(b), u), z)$
x cu $h(z) \Rightarrow \{x \mapsto h(z)\}$
$g(x, s(z))$ cu $g(h(b), u) \Rightarrow$
x cu $h(b) \Rightarrow h(z)$ cu $h(b) \Rightarrow \{x \mapsto h(z), z \mapsto b\}$
$s(z)$ cu $u \Rightarrow \{x \mapsto h(z), z \mapsto b, u \mapsto s(z)\}$
t cu $z \Rightarrow t$ cu $b \Rightarrow \{x \mapsto f(z), z \mapsto b, u \mapsto s(z), t \mapsto b\}$

Union-Find cu dicționare

Substituție = *dicționar* de la variabile (string) la termeni

union(D, T_1, T_2) produce *o nouă substituție* care unifică T_1 cu T_2
pornind de la o substituția / dicționarul D existent
construim substituția pas cu pas, unificând succesiv termeni

find(D, X) caută recursiv reprezentantul lui X ("ce înseamnă")
adică aplică lui X substituțiile din D

Exemplu: unificăm $f(x, g(x, s(z)), t)$ cu $f(h(z), g(h(b), u), z)$
x cu $h(z) \Rightarrow \{x \mapsto h(z)\}$
$g(x, s(z))$ cu $g(h(b), u) \Rightarrow$
x cu $h(b) \Rightarrow h(z)$ cu $h(b) \Rightarrow \{x \mapsto h(z), z \mapsto b\}$
$s(z)$ cu $u \Rightarrow \{x \mapsto h(z), z \mapsto b, u \mapsto s(z)\}$
t cu $z \Rightarrow t$ cu $b \Rightarrow \{x \mapsto f(z), z \mapsto b, u \mapsto s(z), t \mapsto b\}$

Substituind până la capăt:
$\{x \mapsto f(b), z \mapsto b, u \mapsto s(b), t \mapsto b\}$