F.O. Unification

FOL vs prop logic

prop logic

atomic props: p & r, p' ... $p \equiv$ "raining in SF"

Boolean combinations $\neg\alpha$, $\alpha \wedge \beta$, $\alpha \vee \beta$, $\alpha \Rightarrow \beta$, ...

Restricted

FOL syntax

Universe: prescribed set of constants

Variables

eg. U = { boba, coffee, tea, Kiki, cup, bicycle, ...}

Variables: x, y, z, etc. ranging over U JoJo, Boba

Predicates: ~~map from U^n to {T, F}~~

~~eg.~~ subset of U^n

eg. Student = { kiki, JoJo Ana}

for predicate P we write

$P(x)$, if $x \in P$

eg. Student (Kiki)
Student (JoJo)

eg. Likes = {(Kiki, Boba), (Kiki, Tea), (JoJo, Boba)}

more expressive →

Likes (Kiki, Boba)
Likes (JoJo, Boba)
Likes (Kiki, Tea)

Functions: $f: U^n \rightarrow U$

ex. FavoriteDrink(Kiki) = Boba

MostRecentCommonAncestor(Kiki, Baba) = JoJo

* ignore for this class

~~Boolean Combinations of Predicates~~

Quantifiers

$\forall x$ Likes(x, Boba) universal

$\exists x$ Likes(x, Boba) existential

for simplifying assumption for us:

only 1 layer of quantification

Boolean Combinations $\neg\alpha$, $\alpha \wedge \beta$, $\alpha \vee \beta$, $\alpha \Rightarrow \beta$...

Ignore Implicit Quantifiers; i.e. ignoring quantifiers

for $\forall x$ Likes(x, Boba) instead write Likes(x, Boba)

for $\exists x$ Likes(x, Boba) instead write Likes(K_i, Boba)

K_i is a Skolem constant;
as yet
some specific but unspecified member of U,
a symbol not present anywhere else in
U: formulas, variables, predicates etc.

F.O. Unification

Example

knowledge base:

$\forall x \; King(x) \wedge Greedy(x) \Rightarrow Evil(x)$

$King(John)$

$Greedy(John)$

---

query: $Evil(x)$ ≡ who is evil?

$\vdots$

$Evil(John)$

$\vdots$

$\{x/John\}$ "unify x with John"

given a knowledge base (collection of F.O.L sentences)

and a query Q (predicate with unknown variables $x_1, \ldots x_n$)

a solution to the query is a unifier.

unifier: set of ~~assignments~~ substitutions

$\theta = \{x_1/t_1, x_2/t_2, \ldots x_k/t_n\}$ such that replacing each x_i with t_i in Q results in Q evaluating to true

Unification a process (for finding a substitution) that makes two expressions look syntactically identical

The UNIFY algorithm applied to two ~~sentences~~ expressions u & v

UNIFY(u, v) returns a unifier, θ, for them if one exists:

UNIFY$(u, v) = \theta$ such that

SUBST(θ, p) = SUBST(θ, q)

Example

Knowledge Base (KB):
- knows (JoJo, Kiki)
- knows (y, Baba)
- knows (y, Teacher(y))
- knows (x, Eddy)

Q: knows (JoJo, x)

UNIFY (knows (JoJo, x), knows (JoJo, Kiki)) = {x/Kiki}

UNIFY (knows (JoJo, x), knows (y, Baba)) = {x/Baba, y/JoJo}

UNIFY (Q, knows (y, Teacher (y)) = {y/JoJo, x/Teacher (JoJo)}

Bug ①

UNIFY (Knows (Jojo, x),
Knows (x , Eddy)) = fail

Fix { rewrite

UNIFY (Knows (Jojo, x)
Knows (z , Eddy)) = {z/Jojo, x/Eddy}

In general: rename all ~~the~~ variables uniquely

"standardizing apart" ≡ "variable unification"

Bug ② Could be more than 1 unifier

UNIFY (Knows (Jojo, x) , Knows (y, z))

θ_1 = {y/Jojo, x/z}

SUBST (u, θ_1) = Knows (Jojo, ~~Jojo~~ z) = SUBST (v, θ_1)

θ_2 = {y/Jojo , x/Jojo , z/Jojo}

SUBST (u, θ_2) = Knows (Jojo, Jojo) = Subst (v, θ_2)

Observe:

$\theta' = \{z / John\}$

$SUBST(\theta_1, \theta') = \theta_2$

- θ_2 can be obtained from θ_1 via θ'

- θ_1 is more general

THM for every pair of unifiable ~~formulas~~ expressions in FOL there is a unique Most General Unifier (MGU)

- MGU Examples

- pseudocode algorithm

UNIFY(u, v, θ) recursive

idea: traverse u & v ~~while applying~~ comparing sub expressions while updating θ

case θ = failure return failure

case $u = v$, return θ

case x is a variable, ~~return unify~~-var (x, y, θ)

case u is a variable, v is a variable or constant

return $\theta \cup \{u/v\}$

case v is a variable, u is a constant

return $\theta \cup \{v/u\}$

case $u = P(e_1, e_2, \ldots e_k)$ $v = P(e_1', e_2' \ldots e_k')$

(note: same name for predicate P
same # args)

recursively unify e_1 with e_1'
e_2 with e_2'
$\vdots$
e_k with e_k'