lecture 19, the formal language GPL

phil1012 introductory logic

overview

this lecture

- an introduction to the formal language GPL
- the limitations of MPL and the motivation for GPL
- the syntax of GPL
- issues in translation with respect to GPL

learning outcomes

- after doing the relevant reading for this lecture, listening to the lecture, and attending the relevant tutorial, you will be able to:
    - explain what GPL can do that MPL cannot
    - identify well-formed formulas of GPL
    - translate propositions and arguments from English into GPL

required reading

- sections 12.1 and 12.2 of chapter 12

the limitations of MPL

the limitations of MPL

- MPL allows us to attribute properties to individuals. e.g.
    - John is tall
    - Jane is fast
- but it does not allow us to express *relations* between individuals. e.g.
    - John likes Jane
    - Jane does not like John
    - Jane prefers Mark to John

- consider a proposition like this:
    - Bill likes Ben
- we might try to translate it using a glossary like this:
    - LxLx: xx likes
    - aa: Bill
    - bb: Ben
- and a translation like this:
    - (La∧Lb)(La \land Lb)
- but this doesn’t say what we want it to say.

- the predicate ‘likes’ is a two-place predicate: it requires two names to make a proposition.
- to get from MPL to GPL we just add two- and in general nn-place predicates.

- to translate a proposition like this:
  - Bill likes Ben
- we need a glossary like this:
  - $L^2 xy$: x likes y
  - a: Bill
  - b: Ben
- and a translation like this:
  - $L^2 ab$

- to translate a proposition like this:
  - Ben lives in London
- we need a glossary like this:
  - $L^2 xy$: x lives in y
  - b: Ben
  - l: London
- and a translation like this:
  - $L^2 bl$

- to translate a proposition like this:
  - Bill likes someone who lives in London
- we need a glossary like this:
  - $N^2 xy$: x lives in y
  - $L^2 xy$: x loves y
  - $P^1 x$: x is a person
  - a: Bill
  - l: London
- and a translation like this:
  - $\exists y (L^2 by \land P^1 y \land N^2 yl)$

- to translate a proposition like this:
  - Bill does not like everyone who lives in London
- we need a glossary like this:
  - $N^2 xy$: x lives in y
  - $L^2 xy$: x loves y
  - $P^1 x$: x is a person
  - b: Bill
  - l: London
- and a translation like this:
  - $\lnot \forall y ((N^2 yl \land P^1 y) \rightarrow L^2 by)$

predicates in GPL

- predicates in GPL are capital letters with a superscript indicating the number of places.
  - $A^{1}, B^{1}, C^{1}, \ldots, A^{2}, B^{2}, C^{2}, \ldots$
- we leave off the superscript when no confusion will result.
- we do not use I^2 as a two-place predicate. We reserve it for a special purpose. (It is the 'I' in GPLI.)

the syntax of GPL

- the syntax of GPL is just like the syntax for MPL except for the clause for atomic wffs which now looks like this.

1. Wffs of PL are defined as follows:

    1. Where Pn_\underline{P^n} is any nn-place predicate and t1_\underline{t_1} …tn_\underline{t_n} are any terms, the following is a wff:

        Pn_\underline{P^n}t1_\underline{t_1} …t2_\underline{t_2}

- that is, an nn-place predicate followed by any mixture of nn names and/or variables is a well-formed formula.
- wffs of this form are atomic.

order matters

- suppose you want to translate this:
    - Bill is heavier than Mary
- and your glossary looks like this:
    - H2xyH^2 xy: xx is heavier than yy
    - bb: Bill
    - mm: Mary
- then your translation must look like this:
    - H2bmH^2 bm
- and not like this:
    - H2mbH^2 mb
- order matters!

order matters

- suppose you want to translate this:
    - Bill is heavier than Mary
- and your glossary looks like this:
    - H2yxH^2 yx: xx is heavier than yy
    - bb: Bill
    - mm: Mary
- then your translation must look like this:
    - H2mbH^2 mb
- and not like this:
    - H2bmH^2 bm

- the best way to get a feel for translations into GPL is to look at some examples

---

- let’s translate this into GPL:

P1.	Bill is heavier than Mary
P2.	Mary is heavy
C1.	Bill is heavy

- our glossary:
    - H2yxH^2 yx: yy is heavier than xx
    - H1xH^1 x: xx is heavy
    - bb: Bill
    - mm: Mary
- our tranlation:
    - H2bm,H1m,∴H1bH^2 bm, H^1 m, \therefore H^1b

- let’s translate this into GPL:
    - Singapore is between Sydney and London
- our glossary:
    - B3xyB^3 xy: xx is between yy and zz
    - gg: Singapore
    - ss: Sydney
    - ll: London
- our translation:
    - B3gslB^3 gsl

- let’s translate this into GPL:
    - Alfred can solve every puzzle.
- our glossary:
    - P1xP^1 x: xx is a puzzle
    - S2xyS^2 xy: xx can solve yy
    - aa: Alfred
- our translation:
    - ∀x(Px→Sax)\forall x (Px \rightarrow Sax)

- let’s translate this into GPL:
    - Alfred can solve any puzzle
- our glossary:
    - P1xP^1 x: xx is a puzzle
    - S2xyS^2 xy: xx can solve yy
    - aa: Alfred
- our translation:
    - ∀x(Px→Sax)\forall x (Px \rightarrow Sax)

- let’s translate this into GPL:
    - Alfred cannot solve every puzzle.
- our glossary:
    - P1xP^1 x: xx is a puzzle
    - S2xyS^2 xy: xx can solve yy
    - aa: Alfred
- our translation:
    - ¬∀x(Px→Sax)\lnot \forall x (Px \rightarrow Sax)
    - ∃x(Px∧¬Sax)\exists x (Px \land \lnot Sax)

- let’s translate this into GPL:
    - Alfred cannot solve any puzzle.
- our glossary:
    - P1xP^1 x: xx is a puzzle
    - S2xyS^2 xy: xx can solve yy
    - aa: Alfred
- our translation:
    - ∀x(Px→¬Sax)\forall x (Px \rightarrow \lnot Sax)
    - ¬∃x(Px∧Sax)\lnot \exists x (Px \land Sax)

multiple quantifiers in GPL

muliple quantifiers in GPL

- let’s take a close look at GPL formulas with multiple quantifiers

---

- consider the open atomic wff SxySxy.

- suppose we have the following glossary:
    - SxySxy: xx sees yy
- to make a proposition (a closed wff) from SxySxy we must add two quantifiers: one containing xx and one containing yy.
- they can be existential or universal.

- here are all the possible combinations:

1. ∀x∀ySxy\forall x \forall y Sxy
2. ∀y∀xSxy\forall y \forall x Sxy
3. ∃x∃ySxy\exists x \exists y Sxy
4. ∃y∃xSxy\exists y \exists x Sxy
5. ∀x∃ySxy\forall x \exists y Sxy
6. ∃y∀xSxy\exists y \forall x Sxy
7. ∀y∃xSxy\forall y \exists x Sxy
8. ∃x∀ySxy\exists x \forall y Sxy

- what do each of them mean?

- consider:
    -
        - ∀x∀ySxy\forall x \forall y Sxy
- roughly: for every x, and for every y, x sees y.
- dynamically: no matter what you pick first—call it xx—and no matter what you pick second—call it yy—xx sees yy.
- translates: everything sees everything.

- consider:
    -
        - ∀y∀xSxy\forall y \forall x Sxy
- roughly: for every y, and for every x, x sees y.
- dynamically: No matter what you pick first—call it yy—and no matter what you pick second—call it xx—xx sees yy.
- translates: everything sees everything.
- 1 and 2 are equivalent

- consider:
    -
        - ∃x∃ySxy\exists x \exists y Sxy
- roughly: for some x, and for some y, x sees y.
- dynamically: You can pick a thing-call it xx—and then pick a thing —call it yy—such that xx sees yy.
- translates: something sees something.

- consider:
    -
        - ∃y∃xSxy\exists y \exists x Sxy
- roughly: for some y, and for some x, x sees y.
- dynamically: You can pick a thing-call it yy—and then pick a thing —call it xx—such that xx sees yy.
- translates: something sees something.
- 3 and 4 are equivalent.

- consider:
    -
        - ∀x∃ySxy\forall x \exists y Sxy
- roughly: for all x, and for some y, x sees y.
- dynamically: No matter what you pick first—call it xx—you can pick a thing—call it yy—such that xx sees yy.
- translates: everything sees something (not “everything sees something *other than itself*”)

- consider:
  - - ∃y∀xSxy\exists y \forall x Sxy
- roughly: for some y, and for all x, x sees y
- dynamically: You can pick a thing–call it yy—such that no matter what you pick second—call it xx—xx sees yy.
- translates: something is seen by everything
- 5 and 6 are not equivalent

- consider:
  - - ∀y∃xSxy\forall y \exists x Sxy
- roughly: for all y, and for some x, x sees y.
- dynamically: No matter what you pick first—call it yy—you can pick a thing—call it xx—such that xx sees yy.
- translates: everything is seen by something.

- consider:
  - - ∃x∀ySxy\exists x \forall y Sxy
- roughly: for all y, and for some x, x sees y
- dynamically: you can pick a thing–call it xx—such that no matter what you pick second—call it yy—xx sees yy.
- translates: something sees everything.

- considering some more examples may be helpful at this stage

---

- let’s the following into GPL:
  - Everyone has a father.
- our glossary:
  - PxPx: xx is a person
  - FxyFxy: xx is a father of yy
- our translation:
  - ∀x(Px→∃yFyx)\forall x(Px \rightarrow \exists yFyx)

- let’s translate the following into GPL:
  - There is someone who is everyone’s father.
- our glossary:
  - PxPx: xx is a person
  - FxyFxy: xx is a father of yy
- our translation:
  - ∃x(Px∧∀y(Py→Fxy))\exists x (Px \land \forall y(Py \rightarrow Fxy))

- let’s translate the following into GPL:
  - No one lacks a father but not everyone is a father.
- our glossary:
  - PxPx: xx is a person
  - FxyFxy: xx is a father of yy
- our translation:
  - ¬∃x(Px∧¬∃yFyx)∧¬∀x(Px→∃yFxy)\lnot \exists x (Px \land \lnot \exists y Fyx) \land \lnot \forall x(Px \rightarrow \exists yFxy)

- let’s translate the following into GPL:
  - There is no such thing as a hotel that has no rooms.
- our glossary:

- HxHx: xx is a hotel
- RxRx: xx is a room
- HxyHxy: xx has yy

- our translation:
  - ¬∃x(Hx∧¬∃y(Ry∧Hxy))\lnot \exists x (Hx \land \lnot \exists y (Ry \land Hxy))

- let’s translate the following into GPL:
  - A teacher who assigns a problem that has no solution has no students who like her.
- our glossary:
  - PxPx: xx is a problem
  - TxTx: xx is a teacher
  - LxyLxy: xx is a solution of yy
  - AxyAxy: xx assigns yy
  - SxySxy: xx is a student of yy
  - KxyKxy: xx likes yy
- our translation:
  - ∀x∀y((Tx∧Py∧Axy∧¬∃zLzy)→¬∃w(Swx∧Kwx))\forall x \forall y((Tx \land Py \land Axy \land \lnot \exists z Lzy) \rightarrow \lnot \exists w(Swx \land Kwx))

wrapping up

this lecture

- the limitations of MPL and the motivation for GPL
- the syntax of GPL
- issues in translation with respect to GPL

next lecture

- lecture 20, the semantics of GPL and trees for GPL