PHIL 331/MATH 281: Week 3

Polish and reverse Polish notation

1. We have defined a set of wffs using so-called **standard notation** (or: *infix notation*). There is an alternative way of defining a set of wffs using so-called **Polish notation** (or: *prefix notation*), invented by the polish logician Jan Łukasiewicz in 1920. The formation rules are as follows:

   - Every sentence letter is a wff.
   - If φ is a wff then so is '¬φ'.
   - If φ and ψ are wffs then so are '∧φψ', '∨φψ', '→φψ', and '↔φψ'.
   - Nothing else is a wff.

   Examples of wffs: 'P', '¬P', '∧PQ', '∨¬P¬Q', '→∧PQ∨¬P¬Q'

2. Polish notation uses no brackets, but still its wffs can be uniquely decomposed: each complex wff can be decomposed as either '¬φ', '∧φψ', '∨φψ', '→φψ', or '↔φψ' in exactly one way (this needs proof). And for each wff we have a unique structural tree.

3. There is a bijection from the set of wffs in standard notation to the set of wffs in Polish notation. We can define the bijection recursively as follows:

   If φ is a sentence letter, then $f(\varphi) = \varphi$
   If φ = '¬ψ', for some wff ψ, then $f(\varphi) =$ '¬' + $f(\psi)$
   If φ = '(ψ ∧ ψ')', for some wffs ψ and ψ', then $f(\varphi) =$ '∧' + $f(\psi) + f(\psi')$
   If φ = '(ψ ∨ ψ')', for some wffs ψ and ψ', then $f(\varphi) =$ '∨' + $f(\psi) + f(\psi')$
   If φ = '(ψ → ψ')', for some wffs ψ and ψ', then $f(\varphi) =$ '→' + $f(\psi) + f(\psi')$
   If φ = '(ψ ↔ ψ')', for some wffs ψ and ψ', then $f(\varphi) =$ '↔' + $f(\psi) + f(\psi')$

   (Note that this function is well-defined)

   Example: f('(¬A → (B ∨ ¬C))')
   = '→' + f('¬A') + f('(B ∨ ¬C)')
   = '→' + '¬' + f('A') + '∨' + f('B') + f('¬C')'
   = '→' + '¬' + 'A' + '∨' + 'B' + '¬' + f('C')
   = '→¬A∨B¬C'.

   If φ is a wff in standard notation, we might say that $f(\varphi)$ is a **translation** of φ into Polish notation, or that φ is a translation of $f(\varphi)$ into standard notation. So '(A ∧ B)' in standard notation is translated as '∧AB' in Polish notation, and '→¬AB' in Polish notation is translated as '(¬A → B)' in standard notation.

4. There is also so-called **reverse Polish notation** (or: *postfix notation*), invented by an Australian philosopher and computer scientist Charles Hamblin in the 1950s. The formation rules are as follows:

   - Every sentence letter is a wff.
   - If φ is a wff then so is 'φ¬'.
   - If φ and ψ are wffs then so are 'φψ∧', 'φψ∨', 'φψ→', and 'φψ↔'.
   - Nothing else is a wff.

   Examples of wffs: 'P', 'P¬', 'PQ∧', 'P¬Q¬∧', 'PQ∧P¬Q¬∨→'

5. Reverse Polish notation also ensures unique decomposition.

6. There is also a bijection from the set of wffs in standard notation to the set of wffs in reverse Polish notation. '(¬A → (B ∨ ¬C))' maps to 'A¬BC¬∨→'.

7. <u>Exercises</u>

   a. Translate from standard notation into Polish notation:
      i. '¬(A ∨ ¬B)'
      ii. '(C → (¬A ∨ B))'
      iii. '(C ∨ ((B ∧ ¬D) → C))'
      iv. '(P ∧ (¬(Q → R) ∨ S))'

   b. Translate from Polish notation into standard notation:
      i. '∧¬→AB→∨AB¬C'
      ii. '∨∧∨¬A¬BC∧∨AC∨¬C¬A'
      iii. '→→AB→→BC→¬AC'
      iv. '→∨¬→¬PQR↔∧P¬QP'

   c. If we count each statement letter as -1, '¬' as 0, and each of the binary connectives as +1, prove that a formula φ is a wff in Polish notation iff (i) the sum of the symbols of φ sum to -1, and (ii) the sum of the symbols in any proper initial segment of φ is non-negative.

   d. Prove that *f* above is a bijection.

   e. Find and give a recursive definition of the following:
      i. a bijection from the set of wffs in Polish notation to the set of wffs in standard notation
      ii. a bijection from the set of wffs in Polish notation to the set of wffs in reverse Polish notation

Substitution

1. If we take the wff '((A ∨ B) ∨ A)' and replace every occurrence of the sentence letter 'A' by an occurrence of the wff '(P ∧ Q)' we get the wff '(((P ∧ Q) ∨ B) ∨ (P ∨ Q))'. We say that '(((P ∧ Q) ∨ B) ∨ (P ∨ Q))' is a **substitution instance** of '(A ∨ B)'.

   We also say that '((A ∨ B) ∨ A)'('(P ∧ Q)'/'A') = '(((P ∧ Q) ∨ B) ∨ (P ∨ Q))'.

2. In general, a wff φ' is said to be a substitution instance of a wff φ just in case there are sentence letters L_1, …, L_k of φ and wffs φ_1, …, φ_k such that φ' can be obtained from φ by replacing every instance of L_1 by φ_1, every instance of L_2 by φ_2, …, and every instance of L_k by φ_k. We write $\varphi' = \varphi(\varphi_1/L_1, \ldots, \varphi_k/L_k)$.

3. Examples. The following are all substitution instances of '(A ∧ (A → B))':

   a. '(A ∧ (A → B))'
   b. '(B ∧ (B → C))'
   c. '(A ∧ (A → (A ∨ A)))'
   d. '((B ↔ D) ∧ ((B ↔ D) → B))'

4. Exercises

   a. Calculate each of the following:
      i. '(A ∧ B)'('(C → C)'/'A')
      ii. '(P → (P ∧ Q))'('(P ∧ P)'/'Q')
      iii. '¬(A ∧ (A ∧ A))'('(A → A)'/'A')
      iv. '(P ∧ (P ↔ Q))'('(C → C)'/'P', '¬D'/'Q')

   b. Prove that a substitution instance of a wff is always a wff.

   c. Prove that any substitution instance of a substitution instance of a wff φ is a substitution instance of φ.

The Semantics of PC

1. We give meaning to PC by **interpreting** its sentence letters.[1]

2. We do this by assigning to each sentence letter a truth value. We can think of an interpretation as a function from the set of sentence letters to the set of truth values.

   Note that what we assign to a sentence letter is a truth value, not a truth condition or a proposition.

[1] Maybe we should think of PC as a language *schema*, rather than as a language?

3. If L is a sentence letter and I is an interpretation then we use '$|L|_I$' to stand for the truth value that I assigns to L. So if I assigns the value true to the sentence letter 'P', then $|\text{'P'}|_I$ = T. We say that 'P' is **true on I**, or that 'P' **denotes** T on I, or that the **extension** of 'P' on I is T.

4. Often it is sufficient for our purposes to consider **partial interpretations** – interpretations in which truth values are assigned to only a finite subset of the sentence letters (often just one or two).

5. An interpretation determines a truth value for every complex wff φ according to the following recursive rules:

   If φ = '$\neg\psi$', for some wff ψ, then φ is true iff ψ is false
   If φ = '$(\psi \wedge \psi')$', for some wffs ψ and ψ', then φ is true iff ψ is true and ψ' is true
   If φ = '$(\psi \vee \psi')$', for some wffs ψ and ψ', then φ is true iff ψ is true or ψ' is true
   If φ = '$(\psi \rightarrow \psi')$', for some wffs ψ and ψ', then φ is true iff ψ is false or ψ' is true
   If φ = '$(\psi \leftrightarrow \psi')$', for some wffs ψ and ψ', then φ is true iff either ψ and ψ' are both true or ψ and ψ' are both false

   Example: suppose that I is an interpretation on which 'A' is true and 'B' is false. Then '$\neg$A' is ___, '$(A \vee B)$' is ___, '$(\neg A \rightarrow B)$' is ___, and '$\neg(A \leftrightarrow (B \wedge B))$' is ___.

   (Note that this function is well-defined. It would not be if we had not used brackets: '$A \wedge B \vee C$')

6. An interpretation thus determines a function from the set of wffs (both atomic and complex) to the set of truth values.

7. If φ is a wff and I is an interpretation then we use '$|\varphi|_I$' to stand for the truth value that I determines for φ (the truth value of φ under I, the truth value of φ relative to I). So if $|\text{'P'}|_I$ = T and $|\text{'Q'}|_I$ = F, then $|\text{'}(P \vee Q)\text{'}|_I$ = T.

8. On the approach we have taken, we have determined a truth value for every wff in the language without specifying an interpretation for the connectives. But we could have taken the following alternative approach:

   - Assign to '$\neg$' the truth function *f* such that $f(x)$ = T iff x = F.
   - Assign to '$\wedge$' the truth function *f* such that $f(x, y)$ = T iff x = T and y = T.
   - Assign to '$\vee$' the truth function *f* such that $f(x, y)$ = T iff x = T or y = T.
   - Assign to '$\rightarrow$' the truth function *f* such that $f(x, y)$ = T iff x = F or y = T.
   - Assign to '$\leftrightarrow$' the truth function *f* such that $f(x, y)$ = T iff either x = T and y = T or x = F and y = F.
   - If C is a 1-place connective then assign to 'Cφ' the result of applying the truth function assigned to C to the truth value assigned to φ. That is: $|\text{'C}\varphi\text{'}|_I = |\text{'C'}|_I(|\varphi|_I)$.

- If C is a 2-place connective then assign to '(φ C ψ)' is the result of applying the truth function assigned to C to the truth values assigned to φ and ψ. That is: $|\text{`}(\varphi\ C\ \psi)\text{'}|_I = |\text{`}C\text{'}|_I(|\varphi|_I,|\psi|_I)$.

We would end up assigning the same truth values to every wff as we did on the first approach.

9. On either approach we can think of each connective as determining or **expressing** a truth function (hence the name 'truth-functional connectives'). We can represent these truth functions using the following truth tables:

x	'$\neg$'(x)
T	F
F	T

x	y	'$\wedge$'(x, y)	'$\vee$'(x, y)	'$\rightarrow$'(x, y)	'$\leftrightarrow$'(x, y)
T	T	T	T	T	T
T	F	F	T	F	F
F	T	F	T	T	F
F	F	F	F	T	T

In a sense, these are the meanings of the connectives.

10. Each wff determines a truth table. Here is the truth table for '(A $\wedge$ B)':

'A'	'B'	'(A $\wedge$ B)'
T	T	
T	F	
F	T	
F	F	

Each row in the truth table represents a possible interpretation of its sentence letters, and hence to a partial interpretation of the language.

11. There is a **long way** and a **short way** of constructing the truth table for a wff.

Here is the long way for '(A $\vee$ (B $\rightarrow$ $\neg$A))':

'A'	'B'	'$\neg$A'	'(B $\rightarrow$ $\neg$A)'	'(A $\vee$ (B $\rightarrow$ $\neg$A))'

Here is the short way:

(	A	∨	(	B	→	¬	A	)	)

12. Note that '(A ∧ B)' and '(C ∧ D)' have different truth tables (their rows represent different truth assignments).

13. Exercises

    a. If 'A' and 'B' are true and 'C' is false on an interpretation I, what are the truth values of the following wffs on I?
       i. '(A ∨ C)'
       ii. '(¬A ∧ ¬C)'
       iii. '(A ↔ (¬B ∨ C))'
       iv. '((B ∨ ¬C) → A)'
       v. '((B ∨ A) → (B → ¬C))'
       vi. '((B ↔ ¬A) ↔ (A ↔ C))'
       vii. '((B → A) → ((A → ¬C) → (¬C → B)))'

    b. If '(A → B)' is true, what can be deduced about the truth values of the following wffs?
       i. '((A ∨ C) → (B ∨ C))'
       ii. '((A ∧ C) → (B ∧ C))'
       iii. '((¬A ∧ B) ↔ (A ∨ B))'

    c. If '(A ↔ B)' is false, what can be deduced about the truth values of the following wffs?
       i. '(A ∧ B)'
       ii. '(A ∨ B)'
       iii. '(A → B)'
       iv. '((A ∧ C) ↔ (B ∧ C))'

    d. Construct truth tables for the following wffs:
       i. '((¬A ∨ B) → C)'
       ii. '((A ↔ B) → (¬A ∧ B))'
       iii. '((A → B) ∨ ¬A)'
       iv. '((A → B) ∧ A)'
       v. '((A ∨ ¬C) ↔ B))'
       vi. '((A → (B → C)) → (A → B))'
       vii. '((A → (B → C)) → ((A → B) → (A → C)))'