UNA NUEVA NOTACIÓN PARA EL CÁLCULO PROPOSICIONAL

J. M. Lorente Tallada
Universidad de Valencia
Juan.Lorente@uv.es

Abstract: The outputs of a Prolog programm are presented. This programm obtains the postfix representation of formulae with the help of an unique operator: the Sheffer bar. This way is good to grasp visually the semantic nature of logic sentences.

Keywords: Sheffer, Lukasiewicz, proof and representation, propositional calculi, logic tree.

EL núcleo del trabajo no es más que otra forma de representar las fórmulas del cálculo proposicional mediante la fusión de dos representaciones conocidas. Esta fusión consiste en aplicar a una fórmula la representación de Sheffer y posteriormente aplicar al resultado la representación polaca o de Lukasiewicz. En suma, se pretende trabajar con un solo operador lógico y sin el uso de paréntesis para lo cual se recurre a su representación polaca de postfijo. La simbiosis de ambas notaciones permite visualizar el árbol de la fórmula.

Peirce[1] presenta en 1880 su notación de una sola conectiva, la "no disyunción" con signo _ y Sheffer en 1913 el functor "no conjunción". Un claro inconveniente visto desde hoy es el uso y abuso del elevado número de signos de puntuación para discriminar las distintas subfórmulas de una fórmula. Así, además de comas, puntos y comas, comillas y paréntesis, se utilizaron barras, cruces, la doble cruz, o dos cortes en la barra, para indicar un mayor alcance… De hecho el número de cortes o cruces utilizado para discriminar alcances no tiene límite y es tanto más elevado cuanto más compleja sea la fórmula ("The Simplest Mathematics", L. I, parágrafo 12). Presentaciones de este tipo quizás sean la causa del poco eco de esta notación.

El uso que haremos de la notación polaca de postfijo viene motivado precisamente para obviar los alcances de los respectivos operadores iterados y por lo tanto para detectar las subfórmulas de una fórmula facilitando la "lectura mecánica" sin necesidad alguna de signos de puntuación.

El que se utilice postfijo en lugar de prefijo no tiene mas justificación que en los algoritmos que presento se utilice vectores aritméticos asociados a las fórmulas, al modo como lo presenta Khorfague en "Logica y Algoritmos", asignando a cada fórmula bien formada el valor unidad y a cada ocurrencia del operador diádico el valor

[1] Peirce, "A boolian algebra with one constant" (1880).

–1. De esta forma se facilita el detectar si una fórmula es o no fórmula bien formada así como los alcances[2] de cada operador.

El uso de esta notación que es una mezcla de las dos mencionadas es para obtener precisamente una representación arbórea de las fórmulas mediante la cual se podrá detectar visualmente, atendiendo sólo a su forma, si una fórmula problema se trata de una tautología, una contradicción o una contingencia. El árbol correspondiente a la fórmula mitigará las dificultades de la lectura de la misma.

Puede parecer caprichoso este tipo de notación pero más caprichoso aun es el resultado que se obtiene para las distintas fórmulas: aplicándolo a fórmulas del cálculo de enunciados podemos comprobar que las tautologias, representadas mediante notación de peirce_de_postfijo, no son más que ristras de símbolos que tienen la peculiaridad de que se desglosa la ristra en dos mitades idénticas (dos ramas principales idénticas) y cada una de esas fórmulas se ramifica en dos, conteniendo una a una fórmula bien formada y la otra se ramifica a su vez en dos y cada una de las ramas contiene a esa misma fórmula. Esto no es más que una forma de interpretar la definición de verdad (valor 1) dada por Peirce en "The Simplest Mathematics" [4.20- nº8] al expresar con su sistema los signos usados en un álgebra booleana o, lo que es lo mismo, la expresión del principio de tercio excluso mediante el operador de Peirce.

El método "formal" de prueba corresponde por tanto a la búsqueda de una estructura arbórea con estas características. Por otra parte, si se efectuase la búsqueda de una contradicción o bien se realizase el anterior proceso de detección de una tautología haciendo uso del operador de Sheffer, la búsqueda se realizaría sobre un campo más restringido, un árbol reducido exactamente a la mitad conformado por dos ramas una de las cuales corresponde a una fórmula, sea A y la otra rama se subdivide en dos cada una de las cuales contiene a esa misma fórmula A.

Es cierto que se suele estar más familiarizados con la notación de infijo así como también con una mayor riqueza de operadores lógicos. Pero estas notaciones no simplificadas y con redundancias obstaculizan la detección de regularidades y en consecuencia la confección de algoritmos.

Para evitarnos la incómoda tarea de las traducciones entre distintas notaciones así como las correspondientes traducciones de traducciones, se hace uso de unos algoritmos encargados de ese trabajo. Se ha confeccionado en PROLOG y como apéndice se muestra una serie de resultados. Así, se cuenta con un traductor de fórmulas con el que cualquier fórmula problema expresada en cualquier notación que utilice las conectivas estándar será aceptada como *input*. Es decir, la fórmula de la que se desea detectar si se trata o no de una tautologia (contradicción o contingencia) se introduce en la notación de infijo usual y es traducida automáticamente a la notación simbiótica propuesta.

Para abreviar la búsqueda en el árbol correspondiente a la fórmula problema, se simplificá la fórmula siempre que sea posible atendiendo a la ley de doble negación. Es decir, siempre que aparezcan en el árbol cuatro ramas idénticas que se puedan superponer dos a dos, se podrá podar el árbol reduciendo estas cuatro ramas a una sola. La explicación es obvia considerando la definición de "¬". Es decir (¬ A) corresponde a (A_A_↓). Y por la doble negación

[2] El alcance o argumento del functor ubicado en la posición *n* será la anterior fórmula bien formada o, lo que es lo mismo, la menor hilera de posición inmediatamente anterior a *n* que en el vector asociado sume la unidad (de igual forma se detectará el otro argumento).

(¬ ¬ A) equivale a A A ↓ A A ↓ ↓

quedará reducida a A.

Igualmente se realizará transposición de nodos y así como podas de un árbol con la obtención de árboles equivalentes debido a las propiedades conmutativa y asociativa. Esto se verá más adelante.

El sencillo método de prueba que se reduce a simple superposición de ristras de símbolos detectada por *matching* es alcanzable mediante los comodines que se usan en cualquier sistema operativo (DOS, UNIX etc,) y que han sido trasladados a los lenguajes de programación e incluso a cualquier tratamiento de textos para efectuar búsquedas y sustituciones. El método de prueba, en consecuencia, será llevado a cabo con total sencillez en cualquier ordenador puesto que se reduce a efectuar comparaciones y se basa exclusivamente en la forma lógica de la fórmula problema obviando cualquier recurso a interpretaciones así como a reglas de inferencia. Sólo se consideran aspectos espaciales.

Desde un punto de vista más formal la cuestión se reduce a lo siguiente.

Definición de fórmula bien formada[Peirce] (fbf)

1-) Una letra enunciativa es una fbf.

2-) Una fbf seguida de una fbf seguida del símbolo "↓" es fbf.

3-) Sólo es fbf lo estipulado en los puntos anteriores 1 y 2.

Así, p, por (1), es fbf, y q , también por (1), es fbf.

pq↓, por (2) es fbf.

pq↓ pq↓↓ es una fbf como también lo son las fórmulas pq↓ pq↓↓p↓ y ppq↓ pq↓↓↓.

Obviamente esta definición construye las infinitas fórmulas bien formadas correspondientes a todo cálculo de enunciados.

Definición de árbol tautológico[Peirce]:

Un árbol es tautológico si es la representación de una fbf y cumple las siguientes condiciones:

-1-) El árbol tiene dos ramas principales y estas son idénticas.[3]

-2-) Cada una de estas ramas se ramifica en dos, una de las cuales corresponde a una fbf y la otra rama se ramifica a su vez en dos conteniendo cada una de ellas a esa misma fbf.

Dicho de otro modo,

El punto (-1-) indica que el árbol es la representación de la estructura Peirce de Postfijo "XX↓", siendo X una fbf.

El punto (-2-) indica que la fbf "X" tiene la estructura Peirce de Postfijo "PPP↓ ↓" o bien "PP↓P↓" siendo P una fbf. Es decir, una rama corresponde a una fbf y la otra a la negación de esa misma fbf.

Propiedad conmutativa: dos ramas de un árbol son conmutables si penden del mismo nodo.

Así, son ramas conmutables las que tienen la estructura PP↓ y P en (-2-). De igual forma son conmutables P y S en PSAA↓↓SAA↓↓↓↓ que es equivalente a SPAA↓↓ PAA↓↓↓↓

Definición de fórmula tautológica[sheffer]:

Una fórmula es tautológica SII

[3] En los gráficos posteriores se indica que una rama es idéntica a otra encerrándola en un círculo. De esta forma se simplifica la representación.

(-1-) Su representación arbórea corresponde a un árbol tautológico.

(-2-) Su representación arbórea corresponde a un árbol con dos ramas principales idénticas cada una de las cuales se ramifica en dos de modo que:

(-a-) una rama corresponde a una fbf

(-b-) la otra rama corresponde a una fórmula tautológica.

Esta definición recursiva genera, o en su caso reconoce, todas las fórmulas tautológicas del cálculo de enunciados.

Estas definiciones que pueden parecer algo caprichosas no corresponden más que a la representación del principio de *tercio excluso* en sus distintas presentaciones. Y estas son las representaciones arbóreas que corresponden a las fórmulas (P v ¬ P), (P v ¬ P v R), (P v R v ¬ P), (P v S v ¬ P v R), (S v P v ¬ P v R) y no hay más formas diferentes de tautología dada la conmutatividad de la disyunción.

Una fórmula es una contradicción[Peirce] SII su representación corresponde al doble o a la mitad de una tautología[Peirce]. Otra forma de definirlo es indicando que, tras ser simplificada la fórmula, sólo cumple el apartado (2) de la definición de árbol tautológico.

Mediante esa representación en forma normalizada de "Peirce_y_ulterior de Lukasiewicz de Postfijo" se tendrá una presentación arbórea mediante la cual se puede detectar si una fórmula es o no un teorema: El método "formal" de prueba se reduce a contemplar la estructura del árbol:

- Si el árbol consta de dos ramas principales, una de las cuales tiene una fórmula bien formada, sea A, y la otra se ramifica en dos sub-ramas cada una de las cuales contiene esa misma fórmula A, se trata de una contradicción.
- Si el árbol consta de dos ramas idénticas cada una de las cuales es como la mencionada en el párrafo anterior, entonces se trata de una tautología.
- En cualquier otro caso se trata de una contingencia.

. Si se parte de la representación de Sheffer en lugar de la de Peirce, se invertirán las interpretaciones correspondientes a contradicción y tautología por lo que si se pretende detectar una contradicción será más simple partir de la representación de Peirce y si lo que se pretende es detectar una taulogía será más simple utilizar la de Sheffer.

Definición de fórmula bien formada[Sheffer] (fbf)

1-) Una letra enunciativa es una fbf.

2-) Una fbf seguida de una fbf seguida del símbolo "|" es fbf.

3-) Sólo es fbf lo estipulado en los puntos anteriores 1 y 2.

Así, p, por (1), es fbf,

q, también por (1), es fbf.

pq|, por (2) es fbf.

pq|pq|| es una fbf como también lo es la fórmula pq|pq||p| .

Obviamente esta definición construye las infinitas fórmulas bien formadas correspondientes a todo cálculo de enunciados.

Definición de árbol tautológico[Sheffer]:

Un árbol es tautológico si es la representación de una fbf y cumple la siguiente condición:

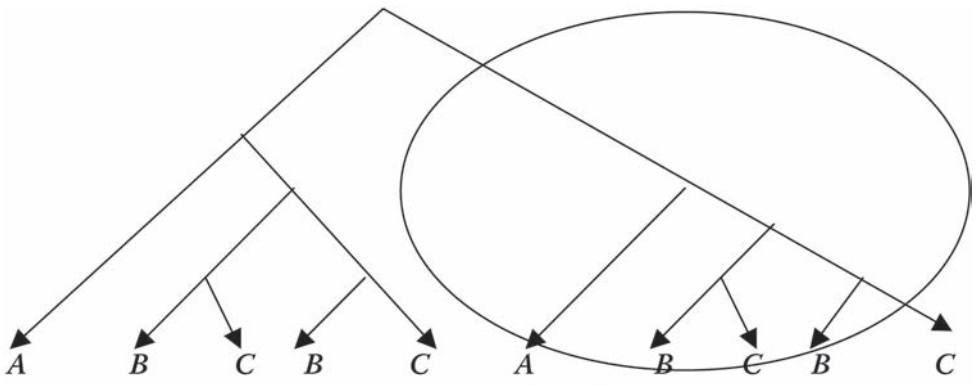
-1-) El árbol tiene dos ramas principales una de las cuales corresponde a una fbf, sea P, y la otra rama se ramifica a su vez en dos conteniendo cada una de ellas a esa misma fórmula P.

Dicho de otro modo, tiene la estructura Sheffer de Postfijo "PPP||" o bien "PP|P|" siendo P una fbf. Es decir, una rama corresponde a una fbf y la otra a la negación de esa misma fbf.

Transposición de nodos y árboles equivalentes

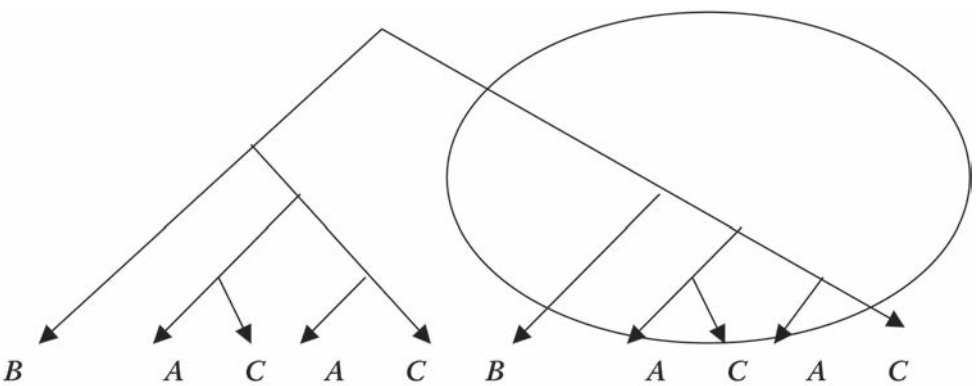
Un árbol es simétrico si se puede desglosar en dos ramas idénticas.

En un árbol simétrico con estructura general ABC|BC|||ABC|BC|||| se puede sustituir distintos contenidos de nodos obteniéndose formulas equivalentes. Así, son equivalentes BAC|AC|||BAC|AC|||| y también CBA|BA|||CBA|BA|||| a la estructura anterior



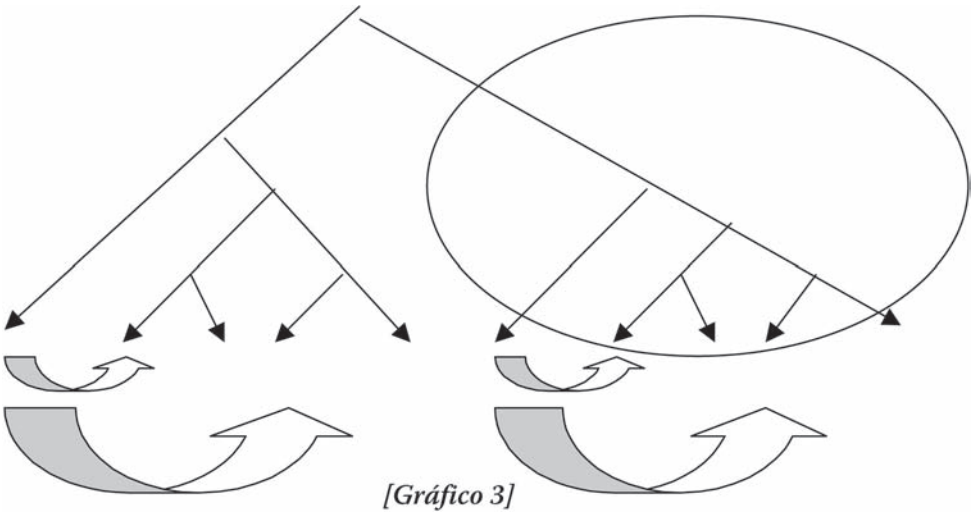
[Gráfico 1]

las dos ramas principales "A" de la figura superior pueden intercambiar el valor del nodo o fbf con las cuatro ramas secundarias ocupadas por el nodo "B" (o alternativamente "C") obteniéndose el árbol equivalente del gráfico 2.



[Gráfico 2]

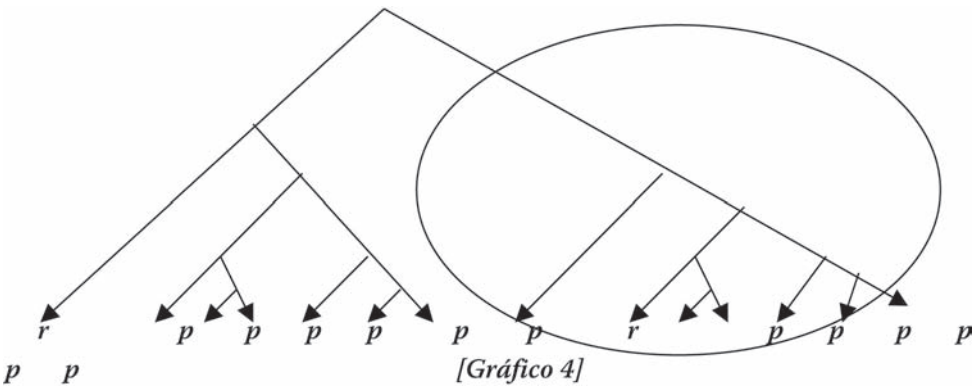
Es decir, puede hacerse el intercambio simultáneo de los nodos señalados por las flechas obteniéndose una fórmula equivalente, y ello por la asociatividad de la disyunción o permutabilidad de ramas de distinto nivel en el operador de Peirce. (Por la propiedad conmutativa, lo mismo podría haberse hecho con la primera y tercera fórmula, es decir, con las dos primitivas ocurrencias de A respecto a las cuatro ocurrencias de C del gráfico 1.)



[Gráfico 3]

Poda de un árbol y obtención de fórmula equivalente:

Un árbol simétrico en el que cada una de las subfórmulas principales contiene en una rama una fbf y en la otra la expresión de una tautología puede ser reducido a sólo la tautología. Así, el árbol



[Gráfico 4]

es equivalente a “p pp| | p pp| | |”, es decir, a la fórmula del gráfico 5.

A continuación presento un conjunto de resultados con la representación propuesta. Corresponden a outputs de un programa en PROLOG que es una ampliación del que se menciona en la nota primera.

§ ?- shefferLukas(‘p > p’).

ppp| |

§ ?- shefferLukas(‘p v ¬ p v r’).

ppp|r|pp|r| | |

```
§ ?- shefferLukas('p v  R v  ¬ p').

  pRpp||Rpp||||


§ ?- shefferLukas('R v  p v  ¬ p')
    Rppp||ppp||||


§ ?- shefferLukas('S v  R v  p v  ¬ p').

  SRppp||ppp||||Rppp||ppp||||||


Y basándonos en la notación alternativa de Peirce

§ ?- peirceLukas('p v  ¬ p').

    1 Izq [p,p,p,§,§]
        2 Izq [p]
        2 Der [p,p,§]
            3 Izq [p]
            3 Der [p]
    1 Der [p,p,p,§,§]
        2 Izq [p]
        2 Der [p,p,§]
            3 Izq [p]
            3 Der [p]

  ppp§§ppp§§§
---------------
  ppp§§
  ppp§§§
yes


§ ?- peirceLukas('p v  ¬ p v  r').


  ppp§r§pp§r§§§ppp§r§pp§r§§§§
---------------
  ppp§r§pp§r§§§
  ppp§r§pp§r§§§§
yes


§ ?- peirceLukas('p v  R v  ¬ p').
```

```
1 Izq [p,R,p,p,§,§,R,p,p,§,§,§,§]
   2 Izq [p]
   2 Der [R,p,p,§,§,R,p,p,§,§,§]
      3 Izq [R,p,p,§,§]
         4 Izq [R]
         4 Der [p,p,§]
            5 Izq [p]
            5 Der [p]
      3 Der [R,p,p,§,§]
         4 Izq [R]
         4 Der [p,p,§]
            5 Izq [p]
            5 Der [p]
1 Der [p,R,p,p,§,§,R,p,p,§,§,§,§]
   2 Izq [p]
   2 Der [R,p,p,§,§,R,p,p,§,§,§]
      3 Izq [R,p,p,§,§]
         4 Izq [R]
         4 Der [p,p,§]
            5 Izq [p]
            5 Der [p]
      3 Der [R,p,p,§,§]
         4 Izq [R]
         4 Der [p,p,§]
            5 Izq [p]
            5 Der [p]

  pRpp§§Rpp§§§§pRpp§§Rpp§§§§§
---------------
  pRpp§§Rpp§§§§
  pRpp§§Rpp§§§§§
yes
```

(Fórmula que es equivalente a la anterior ya que, por conmutatividad, “Rpp§§” es “pp§R§”.)

```
§ ?- peirceLukas('R v  p v  ¬ p').

   1 Izq [R,p,p,p,§,§,p,p,p,§,§,§,§]
      2 Izq [R]
      2 Der [p,p,p,§,§,p,p,p,§,§,§]
         3 Izq [p,p,p,§,§]
            4 Izq [p]
            4 Der [p,p,§]
               5 Izq [p]
```

```
                5 Der [p]
            3 Der [p,p,p,§,§]
              4 Izq [p]
              4 Der [p,p,§]
                5 Izq [p]
                5 Der [p]
    1 Der [R,p,p,p,§,§,p,p,p,§,§,§,§]
      2 Izq [R]
      2 Der [p,p,p,§,§,p,p,p,§,§,§]
        3 Izq [p,p,p,§,§]
          4 Izq [p]
          4 Der [p,p,§]
            5 Izq [p]
            5 Der [p]
        3 Der [p,p,p,§,§]
          4 Izq [p]
          4 Der [p,p,§]
            5 Izq [p]
            5 Der [p]

  Rppp§§ppp§§§§Rppp§§ppp§§§§§
---------------
  Rppp§§ppp§§§§
  Rppp§§ppp§§§§§


§ ?- peirceLukas('S v  R v  p v  ¬ p').


  SRppp§§ppp§§§§Rppp§§ppp§§§§§§SRppp§§ppp§§§§Rppp§§ppp§§§§§§§
---------------
  SRppp§§ppp§§§§Rppp§§ppp§§§§§§
  SRppp§§ppp§§§§Rppp§§ppp§§§§§§§
yes


§ ?- peirceLukas('R v  p v  ¬ p v  S').


  Rppp§S§pp§S§§§ppp§S§pp§S§§§§§Rppp§S§pp§S§§§ppp§S§pp§S§§§§§
---------------
  Rppp§S§pp§S§§§ppp§S§pp§S§§§§§
  Rppp§S§pp§S§§§ppp§S§pp§S§§§§§§
yes


§ ?- peirceLukas('R v  p v  S v  ¬ p').
```

```
  RpSpp§§Spp§§§§pSpp§§Spp§§§§§§RpSpp§§Spp§§§§pSpp§§Spp§§§§§§§
---------------
  RpSpp§§Spp§§§§pSpp§§Spp§§§§§§
  RpSpp§§Spp§§§§pSpp§§Spp§§§§§§§
yes
```

Apéndice

/* El predicado PROLOG shefferLukas( ) ofrece como resultado la mitad de la hilera que ofrece el predicado peirceLukas( ). Es más simple por no tener que presentar la segmentación del árbol. Su estructura es por lo tanto similar a

```
peirceLukas(Hilera):-  hileralukasPeirce(Hilera, _LukasPeirce).
hileralukasPeirce(Hilera, LukasPeirce):- name(Hilera,ListaAscii),
      pasa_Lista_Ascii_A_Lista_Carac(ListaAscii, Infijo),
      polaca(Infijo,[],Pol),        trad_peirce(Pol,1,LukasPeirce).
```

con la diferencia del cambio de operador.