Diferentes Semánticas Declarativas de Lógicas No-Monótonas

Existe una relación entre Programación lógica y razonamiento no-monótono, Primeramente, la programación lógica ofrece un lenguaje de implementación de alto nivel con sabor declarativo. Por lo tanto, no es de sorprendernos, que el primer sistema exitoso real para razonamiento no-monótono, Sistema Theorist Poole's fuerá implementado en Prolog[16].

Pero la relación entre ambos campos va más alla, que el uso de programación lógica como un lenguaje de implementación. Actualmente, la (*NF*), se da en la interpretación de negación en programas lógicos como falla, es probar una fórmula átomica ground, que es un ejemplo de comportamiento no-monótono.

Dado un programa lógico normal,

$A \leftarrow \neg B$,

A es consecuencia, si *B* no se conoce, Pero si sumamos la cláusula

$B \leftarrow$.

en el programa lógico, *A* no es consecuencia. Por expander el programa lógico perdemos una consecuencia.

Además de la cláusula $A \leftarrow \neg B$, si quisieramos saber cuales son modelos, tendríamos todas las combinaciones posibles,

$I_1 = \emptyset$

$I_2 = \{A\}$.

$I_3 = \{B\}$.

$I_4 = \{A,B\}$.

I_2 , I_3 y I_4 modelan para solución, y para saber cuales son modelos mínimos, tenemos que I_2 e I_3 son $\subset$ .. de I_4, pero I_2 e I_3 son modelos mínimos. Como podemos observar en un programa lógico normal puede existir más de un modelo mínimo.

Existen muchas maneras de interpretación con el operador de negación y lógicas no-monótonas que son muy utilizadas en el estudio de semánticas para programas lógicos.

Las primeras y principales extensiones a la semántica lógica con razo-namiento no-monótono son:

1. La Completitud de Clark o ***COMP,*** y

2. Los modelos canónicos:

a. La Semántica Estable o ***STABLE***.

b. La Semántica Bien Fundamentada O ***WFS***(Well Founded Semantics).

c. La Semántica COMP + WFS = ***CWFS.***

3.1 Completitud de Clark

La primera propuesta fue dada a conocer en 1978 y se llamó completitud de Clark[30], la idea principal para deducir información negativa de un programa normal, debemos completar el programa agregándole las mitades de la definición de los simbolos de predicados.

Ejemplo 9: la completitud de $r \leftarrow \neg s$ sería :

$r \leftrightarrow \neg s$

$\neg s$

Observe que $\neg s$ se da para definir la orientación de la cláusula original.

Esto es, $r \leftarrow \neg s$, es lógicamente equivalente $s \leftarrow \neg r$ sin embargo, ambas cláusulas tienen diferente significado y por lo tanto a cada una de ellas se le asocia una semántica diferente.

En algunas ocasiones la completitud de Clark es inconsistente, esto es, puede no tener modelos 2-valuados(cierto o falso), en tal caso la semántica de Clark está indefinida.

Ejemplo 10: la completitud de Clark del Programa siguiente:

$a \leftarrow \neg b, \neg a$

es:

$a \leftrightarrow \neg b, \neg a$

$\neg b$

el cual es inconsistente. Sin embargo si se añade la cláusula $b \leftarrow b$, la completitud llega a ser:

$a \leftrightarrow \neg b, \neg a$

$b \leftrightarrow b$

la cual tiene como modelo a $\{b\}$. Por otro lado, si añadimos, otra cláusula $a \leftarrow a$, la completitud es:

$a \leftrightarrow .. \neg .. b, \neg a$

$a \leftrightarrow a$

la cual tiene como modelo a {a}. Por lo tanto fue necesario encontrar nuevas formas para especificar la semántica de un programa lógico.

3.2 Modelos canónicos

Esta técnica utiliza en un inicio la idea de estratificación de un programa que se ven más a detalle en [22,31].

La semántica *WFS* [16,11], y la semántica *STABLE*[6], son técnicas generales para asignar semánticas a un programa lógico normal, generalizando las técnicas basadas en estratificación.

La semántica *CWFS*, es una semántica propuesta aquí en la udlap por el Dr. Osorio [29,31].

3.2.1 La Semántica Estable o STABLE

La inconsistencia presentada por *COMP* hace necesario encontrar y utilizar otras técnicas. Esta técnica lo constituyen los modelos canónicos que transforman y reducen a una forma canónica los programas sin afectar su semántica, en principio utilizan una forma de generalización de la técnica de estratificación de un programa. Un programa es estratificado si no existe recursión a través de las negaciones. La semántica *STABLE* es la que utiliza la transformación de tres estados para reducir un programa *P* a su forma canónica, se define lo siguiente:

Definición 4[19]: La instanciación de Herbrand de un programa lógico normal es el conjunto de reglas que se obtienen al sustituir los términos del universo de Herbrand por las variables en cualquier combinación posible. Un programa lógico no instanciado se asume que tiene un conjunto finito de reglas, mientras que un programa instanciado puede ser infinito, si existen simbolos funcionales.

Definición 5[19]: Dado un programa lógico *P*, y su instanciación de Herbrand, P_H, definimos *S*, la transformación *STABLE*, la cual va del conjunto de interpretaciones totales al conjunto de interpretaciones totales. Y dada una interpretación total *I*, su transformación *S*(*I*) es definida por los siguientes tres estados:

1. Se define :
   $P' = T_1(P_H, I)$,
   donde T_1 es la siguiente transformación: Para cada regla instanciada, si esta contiene una literal negada la cual es inconsistente con *I*, entonces la regla instanciada es descartada. La

salida de la transformación es el conjunto de reglas instanciadas que no son eliminadas.

2. Se define:
   $P''=T_2(P')$,
   donde T_2, es una transformación mediante la cual todas las literales negativas son eliminadas de P' dejando un programa Horn. Llamaremos a P" la reducción de P'con respecto a *I*.

3. Como P" es un programa Horn, podemos formar sus modelos mínimos de acuerdo a la semántica estándar de Van Emden y Kowalski.

Se define *S*(*I*) como el modelo mínimo de P".

Ejemplo 11:
Sea

$p \leftarrow \neg p$,

$a \leftarrow \neg b$,

$b \leftarrow \neg a$,

$p \leftarrow a$,

y sea {a,¬ .. b,p} el modelo mínimo de P_H. Entonces P' consiste de
$a \leftarrow$ .. ¬ .. b,

$p \leftarrow$ .. ¬ .. a,

ya que la otras reglas contienen variables negadas, cuyos átomos están en *Pos*(*M*). Ahora, "P" esta formado por las reglas Horn:
a
$p \leftarrow$ .. a
así, *S*(M)= {a,¬ .. b,p}, el cual se reproduce de *M*, por lo tanto *M* es un modelo Estable.

Veamos otro ejemplo

Ejemplo 12: Sea,
$a \leftarrow \neg b$,
el modelo Stable es {a}, aunque modelan {a}, {b} y {a,b}, el modelo {a}, es un modelo intended.

La semántica *Stable*, al igual que la completitud de Clark, primero, solo esta definida para una clase restringida de programas y segundo, que no siempre se obtiene la semántica esperada. Ejemplos de semántica Stable

desfavorable los encontramos en [23,19].

3.2.2 Semántica bien Fundamentada (WFS)

Motivación

Porque la Semántica de un modelo Stable no siempre es satisfactoria?, Una razón es que un programa puede tener más de un modelo Stable. cual es el correcto? o podría ser la intersección de todos los modelos Stables?, el problema es que en programación lógica, es común asociar un significado, en donde la representación del conocimiento es más flexible en admitir alternativas. Lo deseable es asignar un único significado para cada programa lógico.

Otro problema crítico de la semántica de *SM* es que un programa lógico *P* puede no tener ningún *SM*; en este caso, ningún significado es asignado a *P*, lo cual no es deseable. Lo peor, es que el fenómeno de no existencia de extensiones es muy irregular, pero que significa, consideremos el siguiente ejemplo.

Ejemplo 13: Sea el programa lógico P_1:

a $\leftarrow$ .. *not b*
b $\leftarrow$ .. *not a.*
P_1 tiene dos modelos Stables, {*a*} y {*b*}. Si sumamos la cláusula
p $\leftarrow$ .. *not p,*
el resultado del programa P_2 no posee ningún modelo Stable, pero si después sumamos la regla p $\leftarrow$ .. a a P_2, el resultado es el programa P_3 y tiene un único modelo Stable {*a, p*}.

Para superar estas y otras dificultades con *SM*, una aproximación alternativa fue dada con las siguientes decisiones[16]:

Exactamente modelo único, llamado el modelo bien fundamentado, esta asociado con cada programa lógico.

Modelo es parcial, significa que los valores de verdad son asignados a una fórmula átomica únicamente, mientras que el resto del programa no es interpretado.

Presentaremos un ejemplo que se apoya en el uso de modelos parciales:

Ejemplo 14:
trabaja $\leftarrow$ .. *not cansado*
dormir $\leftarrow$ .. *not trabaja*
cansado $\leftarrow$ .. *not dormir*
enojado $\leftarrow$ .. *trabajo, not pagar*

pagar ←

las primeras tres reglas describen una interconexión general entre los predicados *trabaja*, *cansado* y *dormir*, pero no proporcionan información sobre cual de ellos es verdadero. De otra manera, *pagar* debe ser siempre verdadero, y *enojado* debe ser falso. Por lo tanto el modelo bien fundamentado del programa es { *pagar*, ¬.. *enojado*}; pero, no proporciona información con respecto a *trabaja*, *cansado* y *dormir*, significa que la interpretación esta abierta, porque no existe información suficiente para decidir, o sea que no existen modelos Stables.

Definición de un Modelo Bien Fundamentado

La siguiente presentación de *WFS* [16], empieza por definir el operador cociente P/M donde *P* es un programa Lógico normal y *M* una interpretación parcial. P/M es obtenido desde *P* por reemplazar en cada cláusula de *P* todos los *Not A* que son verdaderos (falso = **f**; indefinido = **u**) en *M* por **t** .

Es fácil ver que si **t** aparece en la premisa de la regla, puede ser omitida, y si **f** aparece en la premisa de la regla, entonces puede ser borrada, sin cambiar el modelo parcial del programa. Si P/M es No-negativo, entonces tiene un único modelo parcial mínimo.

Teorema 1[16]: Para todo programa lógico *P* y toda interpretación parcial *M*, existe un único modelo parcial mínimo *LPM*(P/M) de P/M.

Consideremos el siguiente ejemplo:

Ejemplo 15: Sea el programa lógico *P*

a ←
b ← .. *not a*
p ← .. *not p.*
y la interpretación parcial {a,¬ .. b}. Entonces, P/M es el programa

a ←
b ← .. **f**
p ← **.. u**.
la segunda regla puede ser borrada porque ocurre **f** en el cuerpo, consideremos el programa lógico no negativo P':

a ←
p ← .. **u**.

El modelo mínimo parcial de P/M' es 1{a,¬ .. b}.

Después de construir el cociente, usamos el *LPM* para asignar un significado a P/M, de esta manera podemos obtener más información sobre la válidez de la fórmula átomica ocurrida en *ground*(*P*), esto

aumenta nuestro conocimieto previo de *M*. De hecho la idea de *WFS* es repetir el proceso mientras es obtenida nueva información , cuando el límite (el punto fijo) es alcanzado, tenemos el modelo bien fundamentado.

Daremos un ejemplo, antes de llegar a la definición formal.

Ejemplo 16: Sea P un programa lógico:

b $\leftarrow$ .. *not a*,
c $\leftarrow$ .. *not b,p*,
p $\leftarrow$ .. *not p*,

inicialmente no tenemos información sobre el programa, entonces comenzamos con la interpretación vacía $M_0 = \emptyset$.. , Ahora construimos el cociente P/M_0 y obtenemos el programa no-negativo,

b $\leftarrow$ .. **u**
c $\leftarrow$ .. **u**, *p*
p $\leftarrow$ .. **u**.

se ve claramente que *LPM*(P/M_0) = {$\neg$ .. a}, tal que, *a* es falso y el resto indefinido, si tomamos la interpretación

M_1 = {$\neg$ .. a} y construimos el cociente P/M_1 :

b $\leftarrow$ .. **t**
c $\leftarrow$ .. **u**, *p*
p $\leftarrow$ .. **u**.
el *LPM*(P/M_1) = {$\neg$ .. a,b} y M_2 = {$\neg$ .. a,b} , repetimos el proceso otra vez, construimos P/M_2:

b $\leftarrow$ .. **t**
c $\leftarrow$ .. **f**, *p*
p $\leftarrow$ .. **u**.

Se borra la segunda regla y obtenemos M_3=*LPM* (P/M_2) = \{\lnot a,b,\lnot c\}$ --> > { .. $\neg$ .. a,b ,$\neg$ .. c}, si repetimos el proceso otra vez, obtenemos exactamente el mismo resultado. Entonces M_3 representa la información máxima que podemos obtener del programa *P* y no es otra cosa que el modelo bien fundamentado de *P*.

Definimos M_0 la interpretación parcial vacía (toda fórmula átomica ground tiene el valor de verdad indefinido). Suponga que M_n es definida para todo $n<k$. Si $k=n+1$ entonces,

M_{n+1}=LPM (P/M),

de otra forma, si *k* es un límite definimos

$M_k = U_{n<k} M_n$

Esta definición es clara para pasos finitos. Daremos otras dos definiciones y un teorema para así concluir lo que respecta a *WFS*.

Definición 6: Modelo parcial Mínimo (*LPM*): Todo programa lógico ground no negativo *P*, tiene un único *LPM*(*P*). Si *P* no es ground, *LPM*(*P*) esta definido como *LPM*(*ground*(*P*)).

Definición 7: El modelo Bien Fundamentado (*WFM*). La secuencia de interpretaciones M_n es definida siempre creciente y tiene un único punto fijo *M* con la propiedad M= LPM(P/M). *M* es llamado *WFM* de *P* denotado por *WFM*(*P*).

Teorema 2: Sea *P* un programa lógico:

1. *WFM*(*P*) es un modelo de *P*.
2. Algún modelo Stable de *P*, extiende a *WFM*(*P*) en el sentido que asigna más valores de verdad a más átomos ground que tenga *WFM*(*P*).
3. Si *WFM*(*P*) es total entonces, es el único modelo Stable de *P*.

lo recíproco de (3) no siempre es cierto.

Ejemplo 17: Sea el programa lógico
p ← .. *not p*,
p ← .. *not q*
q ← .. *not p*.
tiene un único Stable model {*p*}, pero *WFM* es $\emptyset$.

3.2.3 La Semántica CWFS

La semántica declarativa de *CWFS*[34], para todo programa lógico normal *P* un modelo *CWFS* de *P* es un modelo apoyado de *P* que extiende a *WFS*(*P*).

Ejemplo 18:

a:- ¬b.

a:- b.

b:- a,¬d.

d:- *e*.

e:- *d*.

WFS(*P*)= {¬e,¬d}

si extendemos

{¬e,¬d,a¬ .. b} pero, no modela por la regla 3, *b*:- a,¬d , si extendemos de otra forma {¬e,¬ .. d,*a*,b} extiende a *WFS* y además es un modelo apoyado, el que sea un modelo apoyado me indica que el cuerpo de la regla debe ser cierta.

Veamos ahora un ejemplo donde es un modelo, pero no es un modelo apoyado.

Ejemplo 19:

a:- b.

{a,¬b} es un modelo del ejemplo, pero no es un modelo apoyado, debido a que el cuerpo de la regla es negativa.

Después de analizar cada una de las semanticas declarativas no-monótonas (COMP, WFS, STABLE y CWFS) y de ver algunos ejemplos, estamos listos para dar nuestro algoritmo para la obtención de modelos.