Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Explicaciones en argumentación abstracta y grafos declarativos

Tesis de Licenciatura en Ciencias de la Computación

Martin Jedwabny

Director: Ricardo Oscar Rodriguez
Codirector: Madalina Croitoru, Pierre Bisquert
Buenos Aires, 2019

EXPLICACIONES EN ARGUMENTACIÓN ABSTRACTA Y GRAFOS DECLARATIVOS

Argumentación es un modelo de razonamiento abstracto [15], el cual consiste de grafos compuestos por nodos, los cuales representan argumentos que pueden atacarse entre sí. El razonamiento rebatible permite alcanzar conclusiones mas alla de posibles contradicciones, las cuales pueden ser retiradas en la presencia de nueva información. La lógica rebatible es un tipo de lógica tanto proposicional como de primer orden que incorpora preferencias para razonar más allá de posibles contradicciones. Este trabajo se basa en la representación presentada por [18] llamada grafos declarativos, en la cual las reglas rebatibles son modeladas dentro de un grafo, el cual permite razonar mediante razonamiento rebatible usando etiquetas como en sistemas de argumentación. Uno de los grandes problemas con este tipo de estructura y con la lógica en general es que la cantidad de información puede llegar a ser muy grande, de forma que entender las conclusiones se vuelve realmente difícil, sobre todo cuando hay problemas de explosión combinatoria. Nuestro problema central en este trabajo será proveer una metodología teórica para explicar las conclusiones de forma entendible y minimal, priorizando las partes claves que llevan a una conclusión. Esta metodología será aplicable tanto para sistemas de argumentación abstracta como para lógicas rebatibles mediante el uso de grafos declarativos.

Palabras claves: Argumentación, Bases de conocimiento, Lógica rebatible, Grafos declarativos, Semánticas de ranking, Explicaciones.

Índice general

1. INTRODUCCIÓN

La pregunta de investigación que vamos a abordar es la siguiente: *¿Qué es una buena explicación en el contexto de un sistema de razonamiento simbólico basado en grafos dirigidos?* En particular, cuando los grafos son sistemas de argumentación abstracta ó grafos declarativos.

Dentro del campo de la inteligencia artificial, la *argumentación abstracta* [15] es una herramienta de razonamiento diseñada para modelar conocimiento con contradicciones de manera de extraer conclusiones. Concretamente, caracterizan argumentos (unidad de conocimiento) como nodos y la contradicción entre ellos como ejes en grafo dirigidos.

En lógica, el *razonamiento rebatible* [28] permite modelar conocimiento con contradicciones y obtener conclusiones rebatibles, es decir, que pueden cambiar en presencia de conocimiento adicional. En este contexto, el conocimiento es compuesto de hechos y reglas, mientras que las contradicciones son expresadas a través de restricciones negativas.

Mediante este tipo de razonamiento, algunos pasos de inferencia pueden ser priorizados frente a otros mediante preferencias. Las preferencias son relaciones de órden entre reglas ó hechos que permiten razonar en presencia de contradicciones [3, 19].

Notablemente, no hay una forma universalmente válida de razonar de manera rebatible. Una característica inherente del razonamiento rebatible es su dependencia sistemática en nociones abstractas e intuiciones largamente debátidas por expertos en lógica desde hace tiempo [3, 20, 22, 29]. Por ejemplo, ¿puede información dudosa (ni cierta ni falsa) ser usada para contradecir otros pasos de inferencia? (manejo de ambiguedad); ¿pueden distintas cadenas de razonamiento para una declaración ser usadas en combinación para defenderla de otras declaraciones que la contradicen? (defensa en equipo).

Los *grafos declarativos* [19] modelan conocimiento mediante grafos dirigidos bipolares (con dos tipos de ejes, soporte y ataque) de manera de poder razonar de manera rebatible. Como veremos, los nodos de estos grafos (declaraciones) contienen fórmulas lógicas. Los ejes son de dos tipos, los ejes de soporte expresan justificación (consecuencia lógica) mientras que los ejes de ataque expresan contradicción. Estos pueden ser interpretados como una instanciación de sistemas de argumentación dialécticos [12], un tipo especial de sistema de argumentación abstracta donde los argumentos se soportan y atacan entre sí.

Como explicamos, los sistemas de argumentación y grafos declarativos pueden ser vistos como grafos dirigidos que permiten modelar conocimiento y contradicción. En ambos casos, las conclusiones pueden ser obtenidas de distintas maneras, llamadas semánticas de aceptabilidad.

- Las semánticas por conjuntos [15] se basan en caracterizar conjuntos compatibles (sin contradicciones) de unidades de información (nodos del grafo).

- Las semánticas graduales [14] permiten obtener valores dentro de cierto rango para cada nodo del grafo, los cuales representan su fuerza ó peso.

- Las semánticas de ranking [11], las cuales generalizan las graduales, permiten obtener un órden entre los nodos del grafo. Es decir, en vez de buscar qué información es aceptada o cuál es su fuerza, intentan obtener un órden que exprese qué unidades de información son más aceptadas que otras.

Varias semánticas por conjuntos fueron definidas tanto para sistemas de argumentación como para grafos declarativos. En este trabajo, adaptaremos distintas semánticas de ranking de sistemas de argumentación a grafos declarativos.

Luego, mostraremos como esos rankings generados pueden ser utilizados para razonar rebatiblemente mediante los grafos declarativos.

Por otro lado, definiremos un esquema para *explicar* nuestros resultados. Explicación es un término ampliamente usado en el contexto de bases de conocimiento y razonamiento [27, 31]. Aunque no hay pocas interpretaciones para el término, una explicación para una conclusión en un sistema de razonamiento está típicamente dirigida a uno de los siguientes motivos:

- Clarificar: asegurar al receptor que el razonamiento detrás de sus conclusiones es válido.
- Enseñar: transferir conocimiento de cierto mecanismo para que el usuario pueda replicar el razonamiento en otras situaciones y otros contextos.
- Persuadir: convencer al usuario que la conclusión tomada es la mejor en presencia de varias conclusiones válidas.

Usualmente, la literatura puede ser dividida entre explicaciones *estáticas* [27, 31] (i.e. fijas, basadas en justificación [16, 17]) y *dinámicas* [4] (i.e. interactivas, basadas en diálogos):

- Estáticas: todo el conocimiento necesario para la explicación esta disponible desde el comienzo. Esta explicación se manifiesta mediante una estructura de conocimiento que explica la conclusión tomada. Adicionalmente, más evidencia puede ser provista acerca de cómo el razonamiento mismo funciona para clarificar las decisiones intermedias del mecanismo.
- Dinámica: la explicación está basada tanto el conocimiento del sistema como el del receptor. Es decir, el receptor puede pedir información adicional y cuestionar el razonamiento mismo, el cual debe explicarse ante las preguntas, que son las que guían la explicación misma.

En esta tésis, vamos a concentrarnos en explicaciones semi-estáticas, ya que no nos interesará modelar diálogos con el receptor de la explicación que puedan cuestionar el razonamiento mismo. Sino que vamos a proveer una metodología que permita empezar desde una conclusión y expandir el conocimiento a medida que el receptor pida más información sobre ciertos pasos intermedios del razonamiento.

Nuestras intuiciones y marco teórico seguirán el trabajo de [25, 26], en otras palabras, que la investigación de las ciencias cognitivas y sociales indican que los receptores de una explicación esperan explicaciones minimales, ya que cuando la información requerida para justificar una conclusión es enorme, la carga cognitiva de una explicación completa es demasiado grande para entender, y luego una explicación solo debe contener aquello que el receptor pide hasta que lo considere suficiente.

La estructura de esta tésis es la siguiente:

- En la sección 2 (recopilación), introduciremos las nociones preliminares requeridas para esta tésis. Presentaremos los sistemas de argumentación como una herramienta de representación y razonamiento abstracta y semánticas de aceptabilidad por

conjuntos y de orden. Luego, definiremos nuestro lenguaje lógico en el contexto del razonamiento rebatible, *chase* como un procedimiento de saturación de bases de conocimiento que evita redundancias y los grafos declarativos como nuestro sistema de representación del conocimiento instanciado.

- En la sección 3 (original), desarrollaremos semánticas de ranking para grafos declarativos. Adaptaremos principios usuales de estas semánticas a nuestros grafos y propondremos nuevos específicos para nuestros casos. Luego, definiermos semánticas de ranking para grafos declarativos y los compararemos con respecto a los principios ya mencionados.
- En la sección 4 (original), presentaremos nuestras explicaciones en el contexto de sistemas de argumentación y grafos declarativos. Definiremos nuestras explicaciones como una serie de pasos progresivos que aumentan el conocimiento disponible. Finalmente, mostraremos como este procedimiento puede priorizar las partes más importantes de una inferencia mediante semánticas de ranking.

2. PRELIMINARES

2.1. Argumentación abstracta

2.1.1. Sistemas de argumentación

En esta sección, vamos a introducir los sistemas de argumentación abstracta de Dung [15]. Estos permtien representar y razonar acerca de conocimiento con contradicciones sin ninguna restricción acerca del formato del conocimiento mismo.

Los sistemas de argumentación representan el conocimiento como nodos en un grafo dirigido y las contradicciones como ejes del grafo, también llamadas ataques.

Definición 2.1.1. Sistema de argumentación [15]. Un sistema de argumentación (AF) es un par $AF = (A, R)$ que representa un grafo en el que A es el conjunto de argumentos y $R \subseteq A \times A$ son llamados ataques. Decimos que un argumento $a \in A$ ataca $b \in A$ si y solo si $(a, b) \in R$. Además, denotamos $Args(AF) = A$ y $Atts(AF) = R$.

Ejemplo 2.1.2. Considerar el siguiente sistema de argumentación $AF = (A, R)$ de la Figura 2.1, donde el conjunto de argumentos es $A = \{a, b, c, d, e, f\}$ y los ataques son $R = \{(a, c), (b, c), (c, d), (e, b), (f, d)\}$.

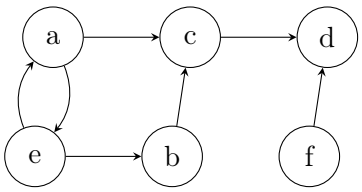

Fig. 2.1: Un ejemplo de un sistema de argumentación

Ejemplo 2.1.3. Los sistemas de argumentación representan los argumentos de forma abstracta, es decir, en principio el contenido de los nodos no tiene estructura. El grafo anterior podría ser un simbolismo para la siguiente discusión:

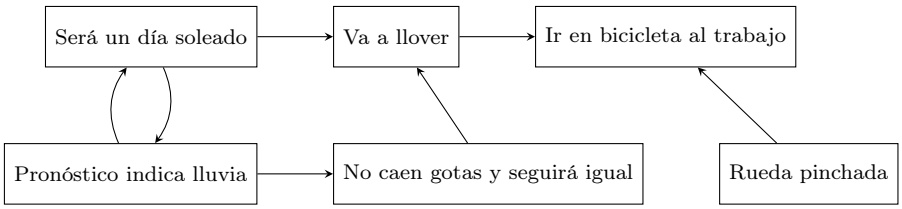

Fig. 2.2: Un ejemplo de un sistema de argumentación

Un subgrafo de un sistema de argumentación es un segundo sistema de argumentación donde todos los argumentos pertenecen al primer sistema y donde se mantienen los ejes, es decir que los argumentos que se atacan son los mismos que se atacan en el primero.

Definición 2.1.4. Subgrafo. Dado un sistema de argumentación $AF = (A, R)$ donde $R \subseteq A \times A$, decimos que $AF' = (A' \subseteq A, R')$ es un subgrafo de AF si y solo si $R' = \{(v_1, v_2) \in R \mid v_1, v_2 \in A'\}$.

Usando la representación de un sistema de argumentación, extendemos la noción de ataque a caminos.

Definición 2.1.5. Caminos, defensores y atacantes. Dado $AF = (A, R)$ un AF y $a, b \in A$. Un **camino** de a a b es un secuencia $\langle a_0 = a, ..., a_n = b\rangle$ tal que $\forall i < n, (a_i, a_{i+1}) \in R$. Entonces, los **atacantes** de b son $R^-(b) = \{a \in A \mid \exists P$ camino de a a b de longitud $|P| \in 2\mathbb{N}_{\geq 0} + 1\}$, los **atacantes directos** de b son $R_1^-(b) = \{a \in A \mid (a, b) \in R\}$, los **defensores** de b son $R^+(b) = \{a \in A \mid \exists P$ camino de a a b de longitud $|P| \in 2\mathbb{N}_{\geq 0}\}$ y los **defensores directos** son $R_1^+(b) = \{a \in A \mid (a, c) \in R$ y $(c, b) \in R\}$.

Ejemplo 2.1.6. (Continuación 2.1.2) Considerar el argumento d, los atacantes directos son $R_1^-(d) = \{c, f\}$, los atacantes $R^-(d) = \{c, f, e\}$ y los defensores (directos) $R^+(d) = R_1^+(d) = \{a, b\}$.

2.1.2. Aceptabilidad por conjuntos

Originalmente, los sistemas de argumentación caracterizan los conjuntos de argumentos aceptables, llamadas exttensiones, según distintas restricciones.

Definición 2.1.7. Conjuntos aceptables y libres de conflictos. Dado un sistema de argumentación $AF = (A, R)$. Un argumento $a \in A$ es **aceptable** con respecto a un conjunto de argumentos $S \subseteq A$ si y solo si S defiende a de todos los ataques, es decir $\forall b \in A$ tal que $(b, a) \in R$, $\exists c \in S$ tal que $(c, b) \in R$. Un conjunto de argumentos $S \subseteq A$ es **libre de conflictos** si y solo si no existen argumentos $a, b \in S$ tal que a ataca b.

Ejemplo 2.1.8. (Continuación 2.1.2) El argumento c es aceptable con respecto a $\{e\}$, el cual es libre de conflictos.

La aceptación de los conjuntos de argumentos definen una semántica. Dos maneras equivalentes de definir estas semánticas son: *basadas en conjuntos* [15] y *basados en etiquetas* [13]. Esta última utiliza tres etiquetas **in**, **out**, y **undec** las cuales representan que un argumento fue aceptado, rechazado o ninguna de las dos, respectivamente.

Definición 2.1.9. Semánticas de aceptación. Dado un sistema de argumentación $AF = (A, R)$, $S \subseteq A$ libre de conflictos con respecto a AF y $F : 2^A \mapsto 2^A$ sea la función definida como $F(S) = \{a \in A \mid a$ es aceptable con respecto a $S\}$.

- S es **admisible** si y solo si $S \subseteq F(S)$.
- S es una extensión **completa** si y solo si $S = F(S)$.
- S es una extensión **grounded** si y solo si S es una extensión completa minimal i.e. $\nexists S' \subset S$ tal que S' es una extensión completa.
- S es una extensión **preferred** si y solo si S es una extensión completa maximal i.e. $\nexists S \subset S'$ tal que S' es una extensión completa.
- S es una extensión **estable** si y solo si S es una extensión preferred tal que S ataca todo argumento en $A - S$.

Ejemplo 2.1.10. (Continuación 2.1.2) Siguiendo el ejemplo, tenemos los siguientes conjuntos:

- $\{a, b, f\}$, $\{a, b\}$, $\{a, f\}$, $\{a\}$, $\{c, e, f\}$, $\{c, e\}$, $\{e, f\}$, $\{e\}$, $\{f\}$ son los conjuntos **admisibles**.

- $\{f\}$ es la única extensión **grounded**.

- $\{a, b, f\}$, $\{c, e, f\}$ son **preferred** y **estables**.

Todas las semánticas definidas hasta ahora pueden tambien ser expresadas a traves de *etiquetas* [13]. Una **función de etiquetado** mapea cada argumento al conjunto de etiquetas $\{in, out, undec\}$:

- **in**: el argumento fue aceptado,

- **out**: el argumento fue rechazado,

- **undec**: ninguna de las dos.

Definición 2.1.11. Etiquetado (válido) [13] Dado un sistema de argumentación $AF = (A, R)$, un etiquetado es una función total $L : A \to \{in, out, undec\}$. Además, L es un etiquetado válido si y solo si:

- $\forall a \in A : L(a) = out$ sii $\exists b \in A$ tal que $(b, a) \in R$ y $L(b) = in$,

- y $\forall a \in A : L(a) = in$ sii $\forall b \in A$ tal que $(b, a) \in R$, $L(b) = out$.

Un sistema de argumentación puede tener distintos etiquetados válidos. También pueden ser vistos como extensiones: los argumentos etiquetados *in* son aceptados por la extensión, *out* es la etiqueta asignada a los argumentos atacados por la extensión, y *undec* son todos los otros argumentos.

Proposición 2.1.12. *Los siguientes conceptos son equivalentes [13]:*

- *(a) extensión completa y (b) etiquetado válido.*

- *(a) extensión grounded, (b) etiquetado válido con conjunto "in" minimal, (c) etiquetado válido con conjunto"out" minimal y (d) etiquetado válido con maximal "undec".*

- *(a) extensión preferred, (b) etiquetado válido con conjunto "in" maximal y (c) etiquetado válido con conjunto "out" maximal.*

- *(a) extensión estable y (b) etiquetado válido con conjunto "undec" vacío.*

Ejemplo 2.1.13. Siguiendo el ejemplo 2.1.2, estos serían los etiquetados que corresponden a las extensiones **estables**. Notar que en este caso el conjunto aceptado es lo más sesgado posible, es decir, que maximiza los argumentos aceptados y minimiza los no aceptados.

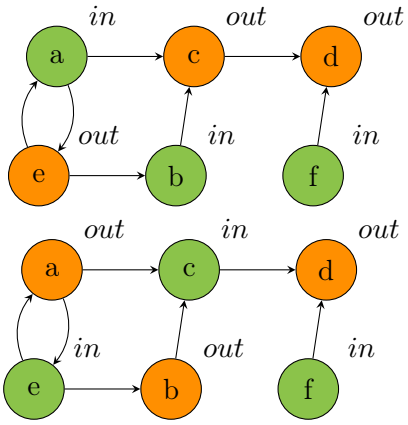

Fig. 2.3: Etiquetado que corresponde a $\{a, b, f\}$ y $\{e, c, f\}$

Y la siguiente figura muestra el etiquetado **grounded**, el cual genera un conjunto "in" minimal, es decir que solo acepta lo que se debe aceptar bajo cualquier circunstancia.

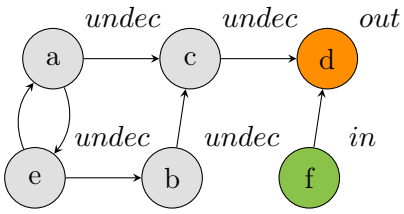

Fig. 2.4: Etiquetado que corresponde a $\{f\}$

2.1.3. Aceptabilidad por órden

Otra manera de caracterizar aceptabilidad es obtener relaciones de orden entre los argumentos, es decir, una relación que indique qué argumento es más aceptable que otro [11] con respecto a distintos criterios. Por favor notar que las siguientes definiciones (2.1.14 a 2.1.21) estarán basadas en [11].

Definición 2.1.14. Semánticas de ranking. Dado un sistema de argumentación $AF = (A, R)$, una semántica de ranking $\leq$ produce un preorden entre los argumentos $\leq_{AF} \subseteq A \times A$ (i.e. una relación binaria reflexiva y transitiva). Denotamos $a <_{AF} b$ sii $a \leq_{AF} b$ y $b \not\leq_{AF} a$. Además, $a \equiv_{AF} b$ sii $a \leq_{AF} b$ y $b \leq_{AF} a$.

Las semánticas de ranking son caracterizadas y analizadas según distintos principios. Estos principios sirven para comparar los distintos rankings según las discrepancias principales de sus mecanismos. Antes de definir ciertos rankings, las siguientes definiciones intermedias serán útiles.

Un orden lexicográfico permite comparar dos listas donde sus elementos son comparables según cierta relación de órden.

Definición 2.1.15. Orden lexicográfico. Dadas dos listas $t = (t_1, ..., t_n), t' = (t'_1, ..., t'_n)$ de números reales, decimos que $t \geq^{lex} t'$ sii $t = t'$ ó $\exists i \leq n$ tal que $t_i > t'_i$ y $\forall j < i$, $t_j = t'_j$.

Una comparación por grupos permite comparar dos conjuntos donde sus elementos son comparables según cierta relación de órden.

Definición 2.1.16. Comparación por grupos. Dado un sistema de argumentación $AF = (A, R)$ y una semántica de ranking $\leq_{AF}$, una comparación por grupos $\leq_{AF}$ es un ranking entre conjuntos de argumentos $S_1, S_2 \subseteq A$ tal que $S_1 \leq_{AF} S_2$ sii existe una función inyectiva $f : S_1 \to S_2$ tal que $\forall a \in S_1, a \leq_{AF} f(a)$.

Un camino es una lista de argumentos donde cada uno ataca el siguiente.

Definición 2.1.17. Adición de caminos. Dados $AF = (A, R), a \in A$. Una adición de camino de defensa para a es $P_+(a) = \langle A', R' \rangle$ tal que $A \cap A' = \{a\}$, $A' = \langle a_0, ..., a_{2n} = a \rangle$ $(n \in \mathbb{N}_{>0})$, $R' = \{(a_i, a_{i+1}) | i < 2n\}$. De forma similar, una adición de un camino de ataque $P_-(a)$ es definido análogamente, pero de longitud impar: $A' = \langle a_0, ..., a_{2n+1} = a \rangle$.

La unión de dos sistemas de argumentación une los argumentos y los ataques de ambos sistemas para formar uno nuevo de la siguiente manera.

Definición 2.1.18. Unión. Dados dos sistemas de argumentación $AF = (A, R), AF' = (A', R')$, $AF \cup AF' = (A \cup A', R \cup R')$.

Dos sistemas de argumentción son isomorfos si son básicamente idénticos pero con distintos nombres para los argumentos, es decir que hay una función inversible que mapea argumentos de un sistema al otro que mantiene los ejes.

Definición 2.1.19. Isomorfismo. Dados dos sistema de argumentación $AF = (A, R)$ y $AF' = (A', R')$, un isomorfismo es una función biyectiva $\lambda : A \to A'$ tal que $\forall x, y \in A, (x, y) \in R \iff (\lambda(x), \lambda(y)) \in R'$.

Definición 2.1.20. Componentes conexas. Las componentes conexas de un AF son los conjuntos de subgrafos maximales (con respecto a inclusión de conjuntos), denotadas $cc(AF)$, donde dos argumentos están en la misma componente de AF si y solo si hay una lista de ejes (ignorando su dirección) entre ellos.

Ahora, definamos principios para caracterizar las semánticas de ranking, bien estudiados en la literatura.

Definición 2.1.21. Principios para semánticas de ranking [11].

- **Abstracción (Abs).** El ranking de un argumento no debería ser definido por su identidad, solo por sus ataques. Dados AF, AF' sistemas de argumentación tales que $\exists \lambda$ isomorfismo de AF hacia AF', entonces $\forall x, y \in Args(AF), x \leq_{AF} y \iff \lambda(x) \leq_{AF'} \lambda(y)$.
- **Independencia (In).** El ranking entre dos argumentos a y b debería ser independiente de cualquier argumento que no esté conectado a a ni b. $\forall AF' \in cc(AF), \forall a, b \in Args(AF'), a \leq_{AF} b \implies a \leq_{AF'} b$.
- **Precedencia Vacía (PV).** Un argumento sin ataques entrantes debería ser tan fuerte como cualquier otro argumento. $\forall a, b \in Args(AF)$, $R_1^-(a) = \emptyset$ y $R_1^-(b) \neq \emptyset$ $\implies b <_{AF} a$.
- **Auto-Contradicción (AC).** Un argumento que se ataca a si mismo debería ser tán débil como cualquier otro argumento. $\forall a, b \in Args(AF), (a, a) \in Atts(AF)$ y $(b, b) \notin Atts(AF) \implies a \leq_{AF} b$.

- **Precedencia de Cardinalidad (PCr).** Un argumento con mayor cantidad de ataques entrantes es más débil que otro con menor cantidad. $\forall a, b \in Args(AF)$, $|R_1^-(a)| < |R_1^-(b)| \implies b \leq_{AF} a$.

- **Precedencia de Calidad (PCl).** Un argumento es más débil que otro si su atacante más fuerte es de mayor ranking que el ataque más fuerte hacia el segundo argumento. $\forall a, b \in Args(AF)$, $\exists c \in R_1^-(a)$ tal que $\forall d \in R_1^-(b), c >_{AF} d \implies a <_{AF} b$.

- **Contra Transitividad (CT).** Si los atacantes directos de un argumento a son al menos tantos y tan fuertes como los de b, entonces b es al menos igual de aceptable que a. $\forall a, b \in Args(AF), R^-(a) \geq_{AF} R^-(b) \implies a \leq_{AF} b$.

- **Contra Transitividad Estricta (CTE).** Si los atacantes directos de un argumento a son más en cantidad y ranking que los de b, entonces b es más aceptable que a. $\forall a, b \in Args(AF), R_1^-(a) >_{AF} R_1^-(b) \implies a <_{AF} b$.

- **Adición de Camino de Defensa (+CD).** Agregar un camino de defensa a cualquier argumento atacado mejora su ranking. Dado un camino de defensa $P_+(a)$, $AF_3 = AF_1 \cup AF_2 \cup P_+(a)$ y $R_1^-(a) \neq \emptyset$, entonces $\lambda(a) <_{AF_3} a$.

- **Incremento de Camino de Defensa (↑CD).** Incrementar la longitud de un camino de defensa de un argumento degrada su ranking. Dado un camino de defensa $P_+(b)$, $b \in R^+(a), b \notin R^-(a), R_1^-(b) = \emptyset$ y $AF_3 = AF_1 \cup AF_2 \cup P_+(b)$, entonces $a >_{AF_3} \lambda(a)$.

- **Adición de Camino de Ataque (+CA).** Agregar an camino de ataque a cualquier argumento degrada su ranking. Dado un camino de ataque $P_-(a)$, $AF_3 = AF_1 \cup AF_2 \cup P_-(a)$, entonces $a <_{AF_3} \lambda(a)$.

- **Incremento de Camino de Ataque (↑CA).** Incrementar la longitud de un camino de ataque de un argumento mejora su ranking. Dado un camino de defensa $P_+(a)$, $b \in R^+(a), b \notin R^-(a), R_1^-(b) = \emptyset$ y $AF_3 = AF_1 \cup AF_2 \cup P_+(b)$, entonces $\lambda(a) >_{AF_3} a$.

- **Total (Tot).** $\forall a, b \in Args(AF)$, $a \leq_{AF} b$ ó $b \leq_{AF} a$.

- **Equivalencia de No-ataque (ENa).** $\forall a, b \in Args(AF)$, $R_1^-(a) = \emptyset$ y $R_1^-(b) = \emptyset$, entonces $a \leq_{AF} b$ y $b \leq_{AF} a$.

- **Precedencia de Defensa (PD).** Dados dos argumentos con la misma cantidad de ataques directos, un argumento defendido es preferido ante uno no defendido. $|R_1^-(a)| = |R_1^-(b)|$, $R_1^+(a) \neq \emptyset$ y $R_1^+(b) = \emptyset$, entonces $a >_{AF} b$.

- **Precedencia de Defensa Distribuida (PDD).** The best defense es when each defender attacks a distinct attacker. Dado AF sea un sistema de argumentación, $|R_1^-(a)| = |R_1^-(b)|$, $|R_1^+(a)| = |R_1^+(b)|$, y:

  - $\forall c \in R_1^+(a)$, hay un único $d \in R_1^-(a)$ tal que $(c, d) \in Atts(AF)$,
  - $\forall c \in R_1^+(b)$, hay un único $d \in R_1^-(b)$ tal que $(c, d) \in Atts(AF)$,
  - $\forall d \in R_1^-(a)$, hay como máximo un único $c \in R_1^+(a)$ tal que $(c, d) \in Atts(AF)$, y

- $\exists d \in R_1^-(b)$, hay dos argumentos $c, c' \in R_1^+(b)$ tal que $(c,d) \in Atts(AF)$ y $(c',d) \in Atts(AF)$.

Entonces $a >_{AF} b$.

- **Ataque vs Defensa Completa (AvsDC).** Si AF es aciclico, $|R_1^-(b)| = 1$, $|R_1^+(b)| = 0$ y $|R^-(a)| = 0$, entonces $a >_{AF} b$.

Ranking Categoriser

Originalmente, Besnard y Hunter [9] propusieron este ranking el cual asigna un número real del intervalo $[0,1]$ a cada argumento a partir de sus atacantes. Aca, un número mayor significa que un argumento es más aceptable. Luego, el ranking es obtenido comparando esos valores.

Definición 2.1.22. Ranking Categoriser [9]. Dado $AF = (A,R)$ un sistema de argumentación. La función categoriser $Cat_{AF} : A \to (0,1]$ es definida como:

$$Cat_{AF}(a) = \begin{cases} 1 & \text{si } \nexists (b,a) \in R \\ \frac{1}{1+\sum\limits_{b \in R_1^-(a)} Cat_{AF}(b)} & \text{en caso contrario} \end{cases}$$

Entonces, dados dos argumentos $a, b \in A$, decimos que:

$$a \leq_{AF}^{Cat} b \iff Cat_{AF}(a) \leq Cat_{AF}(b).$$

Proposición 2.1.23. ***[11]*** *El ranking Categoriser satisface Abs, In, PV, PD, CT, CTE, ↑CD, +CA, ↑CA, Tot y ENa. Los demás principios no son satisfechos.*

Ejemplo 2.1.24. Ranking Categoriser [9]. Dado un sistema de argumentación como el siguiente, el ranking generaría los valores:

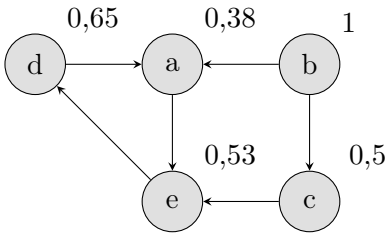

Fig. 2.5: Ejemplo categoriser

Luego, $b >_{AF}^{Cat} d >_{AF}^{Cat} e >_{AF}^{Cat} c >_{AF}^{Cat} a$.

Es importante mencionar que aunque este ranking fue propuesto para argumentos de un sistema de argumentación acíclico, este fue después extendido para cualquier tipo de sistema de argumeción abstracto clásico [30].

Ranking Burden

En [1] Amgoud y Ben-Naim propusieron otra función de peso diferente al categoriser, el cual asigna sucesivos *números de Burden* a cada argumento. En cada paso, el peso de cada argumento se recomputa con respecto a los valores de sus atacantes en las iteraciones previas. Esto produce una lista de números que pueden ser comparados lexicográficamente para determinar las comparaciones entre los argumentos.

De forma intuitiva, esta semántica comienza asignando el valor 1 a cada argumento. Luego, en cada paso, computa un nuevo valor sumando la inversa multiplicativa de cada uno de los valores de los argumentos atacantes en el paso anterior. Esto puede ser interpretado como que los pesos van teniendo en cuenta caminos de longitud superior a medida que la lista se va agrandando.

Finalmente, esto genera una lista de valores para cada argumento. Un argumento es superior a otro si su lista es inferior. En otras palabras, si en algún paso un argumento A tiene mayor valor que B, A se considera peor que B (ya que sumamos la inversa multiplicativa de sus atacantes). Esto significa que aunque en un paso futuro A puede ser superior que B, no significa nada si antes B era superior a A ya que comparamos de forma lexicográfica. Así, se priorizan los caminos de ataque y defensa más cortos.

Definición 2.1.25. Ranking Burden [1]. Dado $AF = (A, R)$ sea un sistema de argumentación. El número Burden en el paso $i \geq 0$ es una función $Bur_i : A \to [1, \infty)$ definida como:

$$Bur_i(a) = \begin{cases} 1 & i = 0 \\ 1 + \sum\limits_{b \in R_1^-(a)} \frac{1}{Bur_{i-1}(b)} & \text{en caso contrario} \end{cases}$$

Que dado un argumento $a \in A$, induce una lista de número de Burden:

$$Bur(a) = \langle Bur_0(a), Bur_1(a), ... \rangle$$

Entonces, dados dos argumentos $a, b \in A$, decimos que:

$$a \leq_{AF}^{Bur} b \iff Bur(b) \leq^{lex} Bur(a).$$

Proposición 2.1.26. ***[11]*** *El ranking Burden satisface Abs, In, PV, PD, CT, CTE, PCr, ↑CA, ↑CD, +CA, Tot, ENa y PDD. Los demás principios no son satisfechos.*

Ejemplo 2.1.27. Ranking Burden [1]. Dado un sistema de argumentación como el siguiente, el ranking generaría los valores:

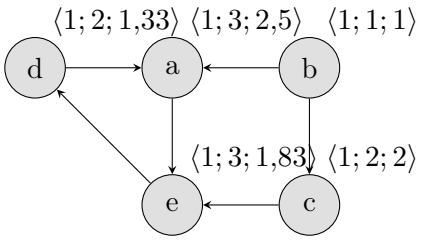

Fig. 2.6: Ejemplo Burden

Luego, $b >_{AF}^{Bur} d >_{AF}^{Bur} c >_{AF}^{Bur} e >_{AF}^{Bur} a$. Notar que el orden entre c y e se invirtió.

Ranking Matt-Toni

Matt y Toni [24] propusieron un método basado en teoría de juegos para las semánticas de ranking. En este, dos jugadores son enfrentados usando un juego de suma zero. Cada jugador debe elegir un subconjunto de argumentos (i.e. las estrategias) para justificar ó contradecir un argumento. Dependiendo qué conjuntos son elegidos por los jugadores, una cierta recompensa es calculada, que indica la fuerza del argumento en caso de elegir esos conjuntos. Dadas todas las posibles formas de elegir esos dos conjuntos aleatoriamente, la fuerza de un argumento es calculada como el valor de este juego de suma zero, el cual busca maximizar la fuerza del argumento desde el punto de vista del primer jugador.

Definición 2.1.28. Ranking Matt-Toni [24]. Dado $AF = (A, R)$ un sistema de argumentación y $a \in A$. Los conjuntos de estrategias de un jugador P y su oponente O para discutir la aceptabilidad de a son $\{P \mid P \subseteq A, a \in P\}$ y $\{O \mid O \subseteq A\}$ respectivamente. Entonces, la recompensa de a con respecto a P y O es:

$$r(P,O) = \begin{cases} 0 & \text{sii } \exists a,b \in P \text{ tal que } (a,b) \in R \\ 1 & \text{sii } \nexists a,b \in P \text{ tal que } (a,b) \in R \ P \text{ y} \\ & \text{no hay ataques de } O \text{ a } P \\ \frac{1}{2}(1 + f(|\{(p,o) \in P \times O \mid (p,o) \in R\}|) & \\ \quad -f(|\{(o,p) \in O \times P \mid (o,p) \in R\}|)) & \text{en caso contrario} \end{cases}$$

Donde la función $f(n) = 1 - \frac{1}{n+1}$.

Es importante que los jugadores elijan una estrategia aleatoria ya que si cierto jugador siempre eligiera la misma, el otro se podría adaptar. Entonces, los autores consideran que ambos jugadores eligen sus estrategias de acuerdo a un vector de probabilidades: $\langle Pr(P_1), Pr(P_2), ...\rangle$ y $\langle Pr(O_1), Pr(O_2), ...\rangle$, entre todas las estrategias mencionadas para cada uno.

Entonces, el peso de cada argumento se determina por el valor caracteristico del juego:

$$MT(a) = \max_{P_i} \min_{O_j} Pr(P_i)Pr(O_j)r(P,O)$$

Finalmente definimos el ranking como:

$$a \leq_{AF}^{MT} b \iff MT(a) \leq MT(b).$$

Ejemplo 2.1.29. Ranking Matt-Toni [24]. Dado un sistema de argumentación como el siguiente, el ranking generaría los valores:

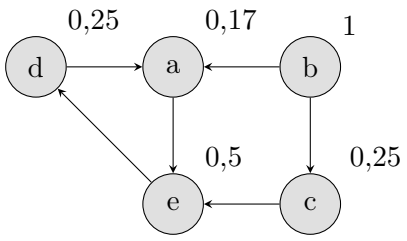

Fig. 2.7: Ejemplo Matt-Toni

Luego, $b >_{AF}^{MT} e >_{AF}^{MT} c, d >_{AF}^{MT} a$. Notar que el orden entre c, d y e es distinto.

Proposición 2.1.30. ***[11]*** *El ranking Matt-Toni satisface Abs, In, PV, +CA, AC, Tot, ENa y AvsDC. Los demás principios no son satisfechos.*

2.2. Grafos declarativos

2.2.1. Lógica de primer órden

Para empezar, introducimos el lenguaje lógico que utilizaremos por el resto de la tésis. Las fórmulas serán definidas de la manera usual, pero solo ciertas formas serán de nuestro interés (i.e. hechos, reglas y restricciones negativas), las cuales serán introducidas a continuación. Además, notar que para expresar incompatibilidad y contradicción utilizaremos restricciones negativas y no un símbolo de negación.

Definición 2.2.1. Lenguaje. Consideramos un lenguaje de lógica de primer orden $\mathcal{L}$ compuesto de (posiblemente infinitas) constantes $\mathcal{C} = \{a, b, ...\}$, variables $\mathcal{V} = \{X, Y, ...\}$, variables "null" $\mathcal{N} = \{Null_1, Null_2, ...\}$, sin símbolos de función adicionales (aparte de las constantes), un conjunto finito de predicados $\mathcal{P}$, y fórmulas construidas con los cuantificadores $(\exists, \forall)$ y el siguiente conector de implicación $(\rightarrow)$ y conjunción $(\wedge)$. Un átomo ó fórmula atómica es $\top, \bot$ (i.e. top, bottom) ó de la forma $p(t_1, \cdot \cdot \cdot, t_k)$, donde p es un predicado y t_i son los términos i.e. $t_i \in \mathcal{C} \cup \mathcal{V} \cup \mathcal{N}$. Dada una fórmula ϕ de $\mathcal{L}$, denotamos $terms(\phi)$ el conjunto de términos y $vars(\phi)$ el conjunto de variables (en $\mathcal{V} \cup \mathcal{N}$) de ϕ.

Un lenguaje lógico es utilizado para representar la realidad de una manera simbólica. An interpretation gives meaning a this symbols by mapping constant symbols y predicados a a new domain $\mathcal{D}$ via a function $\mathcal{I}$.

Definición 2.2.2. Interpretación y Modelo. Una **interpretación** de un lenguaje lógico $\mathcal{L}$ construido sobre un vocabulario $\mathcal{V}oc = (\mathcal{P}, \mathcal{C})$ de predicados y constantes, respectivamente, es un par $(\mathcal{D}, \mathcal{I})$ donde $\mathcal{D}$ es un conjunto no vacío llamado dominio de interpretación y $\mathcal{I}$ es una función de interpretación tal que:

1. $\forall c \in \mathcal{C}$, $\mathcal{I}(c) \in \mathcal{D}$.
2. $\forall p \in \mathcal{P}$ de aridad k, $\mathcal{I}(p) \subseteq \mathcal{D}^k$.
3. $\forall c \neq c' \in \mathcal{C}$, $\mathcal{I}(c) \neq \mathcal{I}(c')$.

Un **modelo** $\mathcal{M}$ de una fórmula ϕ construida sobre $\mathcal{L}$ es una interpretación de $\mathcal{L}$ que hace que esta fórmula sea verdadera al considerar la interpretación clásica de los conectores lógicos y cuantificadores, denotamos esto como $\mathcal{M} \models \phi$.

Uno de los problemas relevantes en las bases de conocimiento es el *entailment problem*, que pregunta si una fórmula es consecuencia de otra(s) fórmula(s).

Definición 2.2.3. Consecuencia Lógica. Dadas dos fórmulas Φ_1 y Φ_2 on $\mathcal{L}$, Φ_2 es una consecuencia (lógica) de Φ_1 (denotado $\Phi_1 \models \Phi_2$) si todos los modelos de Φ_1 son modelos de Φ_2. Denotamos $\Phi_1 \equiv \Phi_2$ si $\Phi_1 \models \Phi_2$ y $\Phi_2 \models \Phi_1$. Dado Γ un conjunto de fórmulas sobre $\mathcal{L}$, $\Gamma \models \Phi_1$ si todos los modelos de Γ son modelos de Φ_1.

Utilizamos hechos para expresar información considerada segura sin necesidad de ningún tipo de presunción.

Definición 2.2.4. Hecho. Un hecho de $\mathcal{L}$ es una fórmula de la forma $\exists\vec{X}\, p(\vec{a}, \vec{X})$, donde $\vec{a}$ y $\vec{X}$ son tuplas de constantes y variables.

Las reglas se utilizan como mecanismo de inferencia para generar nuevo conocimiento.

Definición 2.2.5. Regla. Una regla r es una fórmula $\forall\vec{X}, \vec{Y},\ Body(\vec{X}, \vec{Y}) \rightarrow \exists\vec{Z}$, $Head(\vec{X}, \vec{Z})$ tal que $\vec{X}, \vec{Y}$ son tuplas de variables, $\vec{Z}$ es una tupla de variables (existenciales), $Body(\vec{X}, \vec{Y}), Head(\vec{X}, \vec{Z})$ son conjunciones no vacías de átomos, también denotado $Body(r)$ y $Head(r)$. También lo llamaremos *regla hecho* cuando $Body(r) = \top$.

2.2.2. Saturación de bases de conocimiento

El *entailment problem* describe el problema de responder qué átomos son verdaderos (implicados) por un sistema lógico y de conocimiento dado. Generalmente se expresa mediante consultas, "queries". Se puede ver que las conjuective queries (consultas conjuntivas) preguntan si hay un conjunto de constantes y variables null que hacen que una conjunción de átomos existencialmente cerrada sea una consecuencia del conjunto de hechos.

Definición 2.2.6. Una **conjunctive query** (CQ) es una conjunción de átomos existencialmente cerrada de la forma $Q(\vec{X}) = \exists\vec{Y}\, \Phi(\vec{X}, \vec{Y})$, donde $\vec{X}$ es un conjunto de variables, $\vec{Y}$ es un conjunto de variables existenciales (posiblemente con constantes) y Φ es una conjunción de átomos. Una **boolean conjunctive query** (BCQ) es una conjunctive query de la forma $Q() = \exists\vec{Y}\, \Phi(\vec{Y})$.

Para definir el mecanismo de inferencia de las bases de conocimiento rebatibles, primero tenemos que formalizar las nociones de homomorfismo que permiten instanciar (eliminar variables y cuantificadores de) hechos y reglas.

Definición 2.2.7. Homomorfismo. Un homomorfismo π de un átomo a_1 hacia un átomo a_2 es un mapeo de cada ocurrencia de $X \in vars(a_1)$ por un elemento en $terms(a_2)$. El átomo resultante después del mapeo se denota $\pi(a_1)$. Un homomorfismo π de un conjunto de átomos S a un conjunto de átomos S' es un mapeo de $vars(S)$ a $terms(S')$ tal que $\pi(S) \subseteq S'$.

Dado un conjunto de hechos $\mathcal{F}$ una regla r y un homomorfismo π, una aplicación de regla agrega a $\mathcal{F}$ todas los mapeos aplicables de r utilizando π.

Definición 2.2.8. Aplicación de la regla. Una regla r se dice que es aplicable a un conjunto de hechos $\mathcal{F}$ si hay un homomorfismo π desde $Body(r)$ a $\mathcal{F}$. En ese caso, la aplicación de r a $\mathcal{F}$ de acuerdo a π (denotado $\alpha(\mathcal{F}, r, \pi)$) agrega a $\mathcal{F}$ un conjunto de hechos $\pi^{safe}(Head(r))$ donde π^{safe} asegura que las variables existenciales son reemplazadas con nulos frescos.

Una derivación es una secuencia. de tuplas $< (\mathcal{F}_0, r_0, \pi_0), (\mathcal{F}_1, r_1, \pi_1), ... >$ donde cada $\mathcal{F}_{i+1}$ es el resultado de agregar los resultados de la aplicación de regla $\alpha(\mathcal{F}_i, r_i, \pi_i)$ a el conjunto de hechos $\mathcal{F}_i$.

Definición 2.2.9. Derivación. Dado un conjunto de hechos $\mathcal{F}$ y un conjunto de reglas $\mathcal{R}$, una derivación de $\mathcal{F}$ con respecto a $\mathcal{R}$ es una secuencia (potencialmente infinita) δ de D_i tal que D_i es una tupla $(\mathcal{F}_i, r_i, \pi_i)$ compuesto de un conjunto de hechos $\mathcal{F}_i$, una regla r_i y un homomorfismo π_i desde $Body(r_i)$ a $\mathcal{F}_i$ dónde: $D_0 = (\mathcal{F}_0 = \mathcal{F}, \emptyset, \emptyset)$, y $\mathcal{F}_i =$

$\alpha(\mathcal{F}_{i-1}, r_i, \pi_i)$. Denotamos por $Facts(D_i)$, $Rule(D_i)$, y $Homo(D_i)$ el conjunto de hechos, reglas y homomorfismo de una tupla D_i.

Una *derivación breadth-first* se obtiene al considerar en cada paso "breadth-first" todas las aplicaciones de regla posibles en el actual conjunto de hechos y aplicándolos todos antes de pasar al siguiente paso. Intuitivamente, una vez que vamos al siguiente paso "breadth-first", no podemos aplicar una regla que podría haberse aplicado en un paso anterior de acuerdo con el mismo homomorfismo.

Llamamos la derivación δ **breadth-first** si para todos $i < j$, si $(\mathcal{F}_{i+1} \backslash \mathcal{F}_i) \cap \pi_j(Body(r_j)) \neq \emptyset$ entonces para todos los $k > j$, $\pi_k(Body(r_k)) \not\subseteq \mathcal{F}_i$. Una derivación **exhaustive breadth-first** asegura que todas las reglas se hayan aplicado de acuerdo con todos los homomorfismos posibles.

Una derivación exhaustiva puede ser infinita y puede contener aplicaciones de reglas redundantes, sin embargo, eliminarlas puede hacer que la derivación exhaustiva sea finita.

Un *derivation reducer* es un formalismo utilizado para describir una función que cambia el conjunto de hechos producido en cada paso de una derivación bajo ciertos criterios [6]. Esto es útil porque permite (bajo ciertas condiciones) que las variables cuantificadas de manera universal y existencial se basen de tal manera que la derivación sature el conjunto de hechos en un número finito de pasos.

Definición 2.2.10. Derivation reducer. Dado un conjunto de hechos $\mathcal{F}$ y un conjunto de reglas $\mathcal{R}$, un derivation reducer σ es una función que toma una tupla de aplicación de regla $D_i = (\mathcal{F}_i, r_i, \pi_i)$ en una derivación $\delta = \langle D_0, \ldots, D_i, \ldots \rangle$ de $\mathcal{F}$ con respecto a $\mathcal{R}$ y devuelve una tupla de aplicación de regla $\sigma(D_i) = (\mathcal{F}'_i, r_i, \pi_i)$ tal que $\mathcal{F}'_i \equiv \mathcal{F}_i$.

Luego, el papel del procedimiento *chase* es una aplicación para eliminar reglas que considera redundante. Lo formalizamos a través de derivation reducers por su simplicidad.

Definición 2.2.11. σ-chase. Dado un conjunto de hechos $\mathcal{F}$, un conjunto de reglas $\mathcal{R}$, un derivation reducer σ, y una derivación exhaustive breadth first $\delta = \langle D_0, \ldots, D_i, \ldots \rangle$ de $\mathcal{F}$ con respecto a $\mathcal{R}$: $\sigma\text{-}chase(\mathcal{F}, \mathcal{R}) = \langle \sigma(D_0), \ldots, \sigma(D_i), \ldots \rangle$ y $\sigma(D_i) \in \sigma\text{-}chase(\mathcal{F}, \mathcal{R})$ si y solo si $Facts(\sigma(D_i)) \neq Facts(\sigma(D_{i-1}))$. Un chase es **finite** si hay un paso después del cual no se generan nuevos hechos.

Aplicar chase a un conjunto de hechos de acuerdo con un conjunto dado de reglas genera un conjunto saturado de hechos.

Definición 2.2.12. Conjunto saturado de hechos. Dado un conjunto de hechos $\mathcal{F}$ y un conjunto de reglas $\mathcal{R}$, el conjunto saturado de hechos es: $Facts(\sigma\text{-}chase(\mathcal{F}, \mathcal{R})) = \bigcup_{D \in \sigma\text{-}chase(\mathcal{F}, \mathcal{R})} Facts(D)$.

Un *modelo universal* de un conjunto de hechos y reglas $(\mathcal{F} \cup \mathcal{R})$ es un modelo representativo de todo modelo de $(\mathcal{F} \cup \mathcal{R})$.

Definición 2.2.13. Modelo universal. Dado un conjunto de hechos $\mathcal{F}$ y un conjunto de reglas $\mathcal{R}$, un modelo universal $\mathcal{M}$ de $(\mathcal{F} \cup \mathcal{R})$ es un modelo de $(\mathcal{F} \cup \mathcal{R})$ tal que para todos los modelos $\mathcal{M}'$ de $(\mathcal{F} \cup \mathcal{R})$, hay un homomorfismo de $\mathcal{M}$ a $\mathcal{M}'$.

No siempre es posible obtener el modelo universal (el conjunto de hechos saturado puede ser infinito), sin embargo, si chase es finito, entonces el modelo del conjunto de hechos saturado es un modelo universal. Por lo tanto, las BCQ se puede expresar utilizando la noción de chase.

Teorema 2.2.14. ***Query entailment y Chase [6].*** *Dado un conjunto de hechos $\mathcal{F}$, un conjunto de reglas $\mathcal{R}$, y una BCQ (boolean conjunctive query) Q: si σ-chase$(\mathcal{F}, \mathcal{R})$ es finito entonces $(\mathcal{F} \cup \mathcal{R}) \models Q$ si y solo si $Facts(\sigma\text{-chase}(\mathcal{F}, \mathcal{R})) \models Q$.*

El siguiente teorema establece que las derivaciones son una forma correcta y completa de resolver el BCQ entailment problem.

Teorema 2.2.15. ***Query entailment y derivaciones [6].*** *Dado un conjunto de hechos $\mathcal{F}$, un conjunto de reglas $\mathcal{R}$, y a una BCQ (boolean conjunctive query) Q: $(\mathcal{F} \cup \mathcal{R}) \models Q$ si y solo si existe una derivación (finita) de $\mathcal{F}$ utilizando $\mathcal{R}$ a $\mathcal{F}'$, un conjunto de hechos tal que $\mathcal{F}' \models Q$.*

Por lo tanto, si chase es finito, el conjunto de hechos saturado contendrá la BCQ.

Corolario 2.2.16. Dado un conjunto de hechos $\mathcal{F}$, un conjunto de reglas $\mathcal{R}$, y una BCQ Q: si σ-chase$(\mathcal{F}, \mathcal{R})$ es finito entonces $Q \in Facts(\sigma\text{-chase}(\mathcal{F}, \mathcal{R}))$.

Note que, como mencionamos en la solicitud de definición de regla, todos los tipos de reglas son tratados de la misma manera en el procedimiento de chase. Por lo tanto, la saturación se basará en reglas estrictas, derrotables y derrotadoras indistintamente. Sin embargo, no todos los átomos instanciados contenidos en el conjunto saturado de hechos son considerados justificados por los conjuntos de hechos $\mathcal{F}$ y reglas $\mathcal{R}$. En cambio, será la responsabilidad del marco de razonamiento rebatible definido más adelante, asignar valores de verdad a través de un mecanismo específico (es decir, funciones de etiquetado).

Se han definido muchos tipos de chase diferentes en la literatura hasta ahora, como el Frontier chase [6] y el Skolem chase [23]. Además, query entailment usando chase se ha demostrado que es indecidible en el caso general [8] debido a que chase podría ser infinito. Sin embargo, restricciones sobre el conjunto de reglas bajo el cual query entailment es decidible han sido definidas:

- Finite Expansion Set [5] se puede definir para cada chase, que garantiza que existe un modelo universal y puede generarse usándolo con las reglas dadas.

- Finite Unification Set [6], que identifica reglas bajo las cuales backwards chaining se detiene.

- Greedy Bounded Treewidth Set [7], que asegura que el resultado (y posiblemente infinito) modelo universal tiene una propiedad llamada bounded treewidth.

Para el resto de esta tésis, asumiremos que nuestro conocimiento puede modelarse siguiendo un conjunto de restricciones como se presentó anteriormente, para que podamos garantizar que σ-chase es finito y genera el conjunto saturado de hechos que captura el modelo universal de nuestros conjuntos de hechos y reglas. Sin embargo, como veremos, chase no es esencial para construir los grafos declarativos, sino que estos son una herramienta de razonamiento independiente que podría ser instanciada usando tanto forward como backwards chaining. De todas maneras, usaremos chase para este trabajo ya que facilita la ejemplificación y provee una forma simple de construir el grafo.

2.2.3. Grafos declarativos

Como se discutió antes, el razonamiento rebatible [28] permite modelar bases de conocimiento en las que el estado de aceptabilidad de la información (predicados lógicos) puede cambiar en presencia de información adicional.

Ahora definamos una *base de conocimiento* en nuestro contexto, que será nuestra estructura base de representación de la información.

Definición 2.2.17. Base de conocimiento. Una base de conocimiento es una tupla $\mathcal{KB} = (\mathcal{T}, \mathcal{R}, \mathcal{N})$ dónde:

- $\mathcal{T}$ es un conjunto de reglas hecho i.e. $\top \rightarrow \exists \vec{X}\, p(\vec{a}, \vec{X})$,
- $\mathcal{R}$ es un conjunto de reglas (no hecho) i.e. $Body(r) \neq \top$, y
- $\mathcal{N}$ es un conjunto de restricciones negativas i.e. $\forall \vec{X}\, Body(\vec{X}) \rightarrow \bot$ expresando incompatibilidad.

  Tenga en cuenta que el símbolo $\top$ debe interpretarse como que no se requiere información para inferir la parte derecha de la regla.

Ejemplo 2.2.18. Consideremos el siguiente ejemplo. En este escenario, tratamos de capturar razones para decidir si debemos llevar nuestra bicicleta al trabajo. Considere la base de conocimiento $\mathcal{KB} = (\mathcal{T}, \mathcal{R}, \mathcal{N})$:

- $\mathcal{T} = \{f_1 : \top \rightarrow tiring(bicycle), f_2 : \top \rightarrow sky(clear),\ f_3 : \top \rightarrow forecast(rain),$
  $f_4 : \top \rightarrow fast(bicycle), f_5 : \top \rightarrow free(bicycle)\}$
- $\mathcal{R} = \{r_1 : sky(clear) \rightarrow weather(nice),\ r_2 : \forall \vec{X}.forecast(\vec{X}) \rightarrow weather(\vec{X}),$
  $r_3 : \forall \vec{X}.tiring(\vec{X}) \rightarrow notTake(\vec{X}),\ r_4 : weather(rain) \rightarrow notTake(bicycle),$
  $r_5 : \forall \vec{X}.fast(\vec{X}) \rightarrow take(\vec{X}),\ r_6 : \forall \vec{X}.free(\vec{X}) \rightarrow take(\vec{X})\}$
- $\mathcal{N} = \{\forall \vec{X}.take(\vec{X}) \wedge notTake(\vec{X}) \rightarrow \bot,\ weather(nice) \wedge weather(rain) \rightarrow \bot\}$

Hay razones para creer que debemos tomar la bicicleta y todo lo contrario. Por ejemplo, se podría decir que como el pronóstico indica lluvia $[f_3]$, eso significa que probablemente llueva $[r_2]$, así que probablemente no deberíamos tomar la bicicleta $[r_4]$. Sin embargo, el cielo siendo *clear* $[f_2]$ nos hace creer que tal vez no llueva $[r_1]$. Además, la bicicleta es *fast* $[f_4]$, y eso nos dice que podría ser una buena opción $[r_5]$.

Para caracterizar y razonar con bases de conocimientos, Hecham et al. [19] introdujeron los *grafos declarativos*, una estructura basada en grafos que contiene fórmulas lógicas. Pueden interpretarse como instancias de Abstract Dialectical Frameworks [12], un formalismo usado para representar sistemas de argumentación [15] (como se ve en la sección 2.1.1) en el que los argumentos pueden apoyarse y atacarse mutuamente. **Importante:** si bien en el trabajo original, las reglas podían ser de tres tipos, en esta tésis solo consideraremos un tipo de regla que debe ser interpretada como el tipo de regla rebatible.

Definición 2.2.19. Declaración. Una declaración $s = (\Phi \rightarrow \psi)$ expresada sobre $\mathcal{L}$ contiene una fórmula lógica en $\mathcal{L}$ dónde Φ es una conjunción de átomos instanciados y ψ es un átomo instanciado. Nosotros llamamos s una **declaración query** si y solo si Φ es una query y $\psi = \emptyset$.

Además, las **declaraciones generadas por** un procedimiento de saturación σ-*chase* de la base de conocimiento $\mathcal{KB} = (\mathcal{T}, \mathcal{R}, \mathcal{N})$ expresada en $\mathcal{L}$, son $s = (\Phi \rightarrow \psi)$ dónde:

- s representa una aplicación de regla $\alpha(\mathcal{F} \subseteq \mathcal{F}^*, r \in \mathcal{T} \cup \mathcal{R}, \pi)$,
- $\mathcal{F}^*$ es el conjunto saturado de hechos de $\sigma\text{-}chase(\{\top\}, \mathcal{T} \cup \mathcal{R})$,
- $\pi \in Homo(D_i \in \sigma\text{-}chase(\{\top\}, \mathcal{T} \cup \mathcal{R}))$, y
- $\Phi = \pi(Body(r))$, $\psi = \pi(Head(r))$.

Dada una declaración $s = (\Phi \to \psi)$ denotamos $Rule(s) = \Phi \to \psi$, $Premise(s) = \phi$ y $Conc(s) = \psi$.

Ejemplo 2.2.20. Siguiendo el ejemplo 2.2.18, $\sigma\text{-}chase$ podría saturar el conjunto de hechos como se muestra a continuación:

$$
\begin{aligned}
&\sigma\text{-}chase(\mathcal{F}, \mathcal{R}) = \\
&\quad <(\mathcal{F}_0 = \{\top\}, \emptyset, \emptyset), \\
&\quad (\mathcal{F}_1 = \mathcal{F}_0 \cup \{\top \to tiring(bicycle)\}, \top \to tiring(bicycle), \emptyset), \\
&\quad (\mathcal{F}_2 = \mathcal{F}_1 \cup \{\top \to sky(clear)\}, \top \to sky(clear), \emptyset), \\
&\quad (\mathcal{F}_3 = \mathcal{F}_2 \cup \{\top \to forecast(rain)\}, \top \to forecast(rain), \emptyset), \\
&\quad (\mathcal{F}_4 = \mathcal{F}_3 \cup \{\top \to fast(bicycle)\}, \top \to fast(bicycle), \emptyset), \\
&\quad (\mathcal{F}_5 = \mathcal{F}_4 \cup \{\top \to free(bicycle)\}, \top \to free(bicycle), \emptyset), \\
&\quad (\mathcal{F}_6 = \mathcal{F}_5 \cup \{sky(clear) \to weather(nice)\}, sky(clear) \to weather(nice), \emptyset), \\
&\quad (\mathcal{F}_7 = \mathcal{F}_6 \cup \{forecast(rain) \to weather(rain)\}, \\
&\quad \forall \vec{X}.forecast(\vec{X}) \to weather(\vec{X}), \{\vec{X} \leftarrow rain\}), \\
&\quad (\mathcal{F}_8 = \mathcal{F}_7 \cup \{tiring(bicycle) \to notTake(bicycle)\}, \\
&\quad \forall \vec{X}.tiring(\vec{X}) \to notTake(\vec{X}), \{\vec{X} \leftarrow bicycle\}), \\
&\quad (\mathcal{F}_9 = \mathcal{F}_8 \cup \{weather(rain) \to notTake(bicycle)\}, \\
&\quad weather(rain) \to notTake(bicycle), \emptyset), \\
&\quad (\mathcal{F}_{10} = \mathcal{F}_9 \cup \{fast(bicycle) \to take(bicycle)\}, \\
&\quad \forall \vec{X}.fast(\vec{X}) \to take(\vec{X}), \{\vec{X} \leftarrow bicyle\}), \\
&\quad (\mathcal{F}_{11} = \mathcal{F}_{10} \cup \{free(bicycle) \to take(bicycle)\}, \\
&\quad \forall \vec{X}.free(\vec{X}) \to take(\vec{X}), \{\vec{X} \leftarrow bicycle\}) >
\end{aligned}
$$

Luego, las declaraciones generadas $\mathcal{V}$ pueden recuperarse tomando los homomorfismos utilizados en $\sigma\text{-}chase$ como se explicó antes:

$$
\begin{aligned}
\mathcal{V} = &\{\top \to tiring(bicycle), \\
&\top \to sky(clear), \\
&\top \to forecast(rain), \\
&\top \to fast(bicycle), \\
&\top \to free(bicycle), \\
&sky(clear) \to weather(nice),
\end{aligned}
$$

$$forecast(rain) \rightarrow weather(rain),$$
$$tiring(bicycle) \rightarrow notTake(bicycle),$$
$$weather(rain) \rightarrow notTake(bicycle),$$
$$fast(bicycle) \rightarrow take(bicycle),$$
$$free(bicycle) \rightarrow take(bicycle)\}$$

Como discutimos antes, usamos restricciones negativas para definir el conflicto. A fin de formalizar los ataques entre declaraciones, definamos primero la noción de conflicto.

Definición 2.2.21. Conflicto de hechos. Dos hechos f_1 y f_2 están en conflicto si el cuerpo de una restricción negativa se puede asignar a $\{f_1, f_2\}$.

Ahora ampliaremos la noción de conflicto entre hechos para formalizar los ataques entre declaraciones, así como definir cuando una declaración soporta (es decir, se usa para inferir el estado de) otra declaración.

Definición 2.2.22. Ataque y Soporte. Dadas dos declaraciones s_1 y s_2:

- s_1 soporta s_2 si y solo si $Conc(s_1) \neq \emptyset$, $Conc(s_1) \in Premise(s_2)$. (Decimos que s_1 soporta s_2 sobre $Conc(s_1)$).
- s_1 ataca s_2 si y solo si $\exists f \in Premise(s_2)$ tal que f y $Conc(s_1)$ están en conflicto. (Decimos que s_1 "undercuts" s_2 en f).

Ejemplo 2.2.23. En nuestro ejemplo actual 2.2.18, $(\top \rightarrow sky(clear))$ soporta $(sky(clear) \rightarrow weather(nice))$ y $(sky(clear) \rightarrow weather(nice))$ ataca $(weather(rain) \rightarrow notTake(bicycle))$ porque la premisa de la segunda declaración está en conflicto con la conclusión de la primera con respecto a la restricción negativa $weather(nice) \wedge weather(rain) \rightarrow \bot$ definida anteriormente.

Ahora podemos juntar las definiciones anteriores y definir nuestra estructura de grafos. Aquí, los ejes son dirigidos y de dos tipos posibles (i.e. soporte y ataque).

Definición 2.2.24. Grafo de declaraciones. Un grafo de declaraciones expresado en $\mathcal{L}$, es un grafo dirigido $\mathcal{SG} = (\mathcal{V}, \mathcal{E}_S, \mathcal{E}_A)$:

- $\mathcal{V}$ es un conjunto de declaraciones.
- $\mathcal{E}_S \subseteq \mathcal{V} \times \mathcal{V}$ es el conjunto de ejes de soporte. Hay un eje de soporte $e = (s_1, s_2) \in \mathcal{E}_S$ si y solo si s_1 soporta s_2.
- $\mathcal{E}_A \subseteq \mathcal{V} \times \mathcal{V}$ es el conjunto de ejes de ataque. Hay un eje de ataque $e = (s_1, s_2) \in \mathcal{E}_A$ si y solo si la declaración s_1 ataca s_2.

Por otra parte, un **grafo de declaraciones generado por** $\mathcal{KB} = (\mathcal{T}, \mathcal{R}, \mathcal{N})$ expresado en $\mathcal{L}$ y una saturación chase *σ-chase*, es $\mathcal{SG}_{\mathcal{KB}} = (\mathcal{V}, \mathcal{E}_S, \mathcal{E}_A)$ como se describió anteriormente, donde $\mathcal{V}$ es el conjunto de declaraciones generadas a partir de $\mathcal{KB}$ y la adición (opcional) de declaraciones query.

Ejemplo 2.2.25. Consideremos la figura 2.8 con el grafo de declaraciones para el ejemplo 2.2.18 con la adición de la query $Q = (take(bicycle) \rightarrow \emptyset)$.

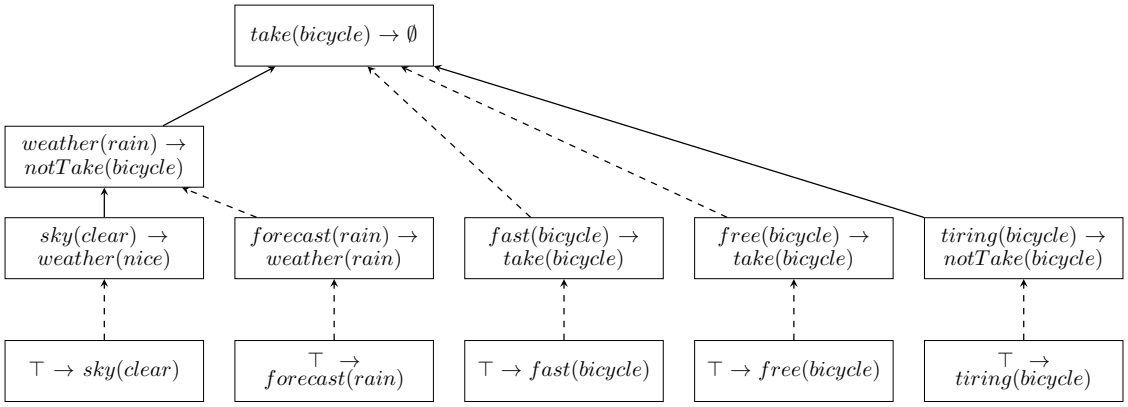

Fig. 2.8: $\mathcal{SG}_{\mathcal{KB}}$ del ejemplo 2.2.18.

Notación 2.2.26. Dado un grafo de declaraciones como en la definición, $Statements(\mathcal{SG}) = \mathcal{V}$. Para un eje $e = (s_1, s_2)$, denotamos s_1 como $Source(e)$ y s_2 como $Target(e)$. Para una declaración s denotamos sus ejes como $\mathcal{E}_A^-(s)=\{e \in \mathcal{E}_A | Target(e) = s\}$, $\mathcal{E}_S^-(s)=\{e \in \mathcal{E}_S | Target(e) = s\}$, $\mathcal{E}_A^+(s)=\{e \in \mathcal{E}_A | Source(e) = s\}$ y $\mathcal{E}_S^+(s)=\{e \in \mathcal{E}_S | Source(e)=s\}$. Los nodos adyacentes entrantes se denotan $\mathcal{V}_A(s)=\{s' \in \mathcal{V} | (s', s) \in \mathcal{E}_A\}$, $\mathcal{V}_S(s)=\{s' \in \mathcal{V} | (s', s) \in \mathcal{E}_S\}$. Decimos que un eje **e es superior a** otro eje $\mathbf{e}'$ y que $\mathbf{e}'$ es inferior a **e** si y solo si $Rule(Source(\mathbf{e})) \succ Rule(Source(\mathbf{e}'))$.

Definición 2.2.27. Soporte completo. Un soporte completo para una declaración s es un conjunto de ejes de soporte denotado $\mathcal{E}_{CS}^s$ tal que:

- $\forall f \in Premise(s)$, $\exists e \in \mathcal{E}_{CS}^s$ tal que $Source(e)$ soporta s en f.
- $\nexists S'$ tal que $S' \subset \mathcal{E}_{CS}^s$ y S' es un soporte completo para s (minimalidad con respecto a inclusión de conjuntos).

Decimos que una declaración $s = (\top \to \psi)$ que contiene una regla de hecho siempre tiene un soporte completo $\mathcal{E}_{CS}^s = \emptyset$.

Ejemplo 2.2.28. En nuestro ejemplo corriente 2.2.18, los conjuntos de declaraciones singleton $\{(fast(bicycle) \to take(bicycle))\}$ y $\{(free(bicycle) \to take(bicycle))\}$ son soporte completo para la declaración query $(take(bicycle) \to \emptyset)$, que solo tiene una premisa.

Antes de definir el estado de aceptabilidad de las declaraciones mediante etiquetas, debemos considerar la presencia de ciclos. Hay dos tipos de ciclos, de soporte (i.e. solo contiene ejes de soporte), y de ataque (i.e. contiene ataques y posiblemente ejes de soporte).

Definición 2.2.29. Ciclo. [18] Un ciclo es una secuencia de ejes $\langle e_0, ..., e_n \rangle$ donde $\forall i,\ e_i \in \mathcal{E}_S \cup \mathcal{E}_A$, $\forall i < n$, $Target(e_i) = Source(e_{i+1})$, y $Source(e_0) = Target(e_n)$.

- **Ciclo de soporte:** un ciclo que solo contiene soportes, $\forall i,\ e_i \in \mathcal{E}_S$.
- **Ciclo de ataque:** un ciclo que contiene al menos un ataque, $\exists i,\ e_i \in \mathcal{E}_A$.

Permitir ciclos de soporte y ataque es posible como se muestra en [18]. Para darles sentido, uno puede considerar *failure-by-looping* [21], un mecanismo para evitar extraer irracionalmente conclusiones en presencia de ciclos.

Para definir correctamente el estado (i.e. etiquetas) de las declaraciones en el grafo, no permitiremos razonamientos circulares y/o ciclos de ataque. Más bien, asumiremos un mecanismo de manejo del ciclo se proporciona como en [18]. En pocas palabras, podemos definir la etiqueta de una declaración s según $\mathcal{V}_S$ y $\mathcal{V}_A$ not considering any statement that belongs to a cycle.

2.2.4. Aceptabilidad por conjuntos para razonamiento rebatible

El razonamiento rebatible permite representar información contradictoria. Para razonar más allá de estas contradicciones, este tipo de razonamiento permite tomar en cuenta preferencias entre las reglas. Luego, es posible decidir qué declaraciones son válidas y cuales no de acuerdo a estas preferencias.

Definición 2.2.30. Relación de preferencia. Dado un conjunto de declaraciones, una relación de preferencias $\prec$ es una relación binaria, reflexiva, transitiva y acíclica de superioridad. Luego, dadas dos declaraciones s, s' decimos que s es superior a s' si y solo si $s' \prec s$.

Luego, mediante preferencias, podemos definir aceptabilidad mediante etiquetados tal como para los sistemas de argumentación de manera de capturar las distintas intuiciones de la lógica rebatible.

Definición 2.2.31. Función de etiquetado. Un etiquetado aplicado a un grafo de declaraciones $\mathcal{SG} = (\mathcal{V}, \mathcal{E}_S, \mathcal{E}_A)$ y una relación de preferencia $\succ$ es una función:

$$L : \mathcal{V} \to \{in, out, ambig, unsup\}$$

Toma como entrada una declaración $s \in \mathcal{V}$ y devuelve una etiqueta. La intuición detrás de estas etiquetas es la siguiente:

1. **in**: indica que la declaración se acepta y su regla se puede aplicar en base a premisas aceptadas.
2. **out**: indica que la declaración no se acepta porque su regla o premisas han sido derrotadas.
3. **ambig**: indica que la regla o las premisas de la declaración son atacadas y la relación de superioridad no puede usarse para determinar si se aceptan o no.
4. **unsup**: indica que al menos una de las premisas de la declaración no está respaldada por otra declaración.

Como se mencionó en la introducción, no existe una forma universalmente válida de razonar rebatiblemente. En lugar, el razonamiento rebatible se basa en un conjunto de intuiciones llamadas *manejo de ambigüedades* y *defensa en equipo*, que son independientes y se ocupan de problemas diferentes.

Para empezar, el *manejo de ambigüedades* se puede abordar de dos maneras diferentes: *bloqueo de ambigüedad* evita que información dudosa (i.e. declaraciones ambiguas) de atacar otras declaraciones, mientras que la *propagación de la ambigüedad* permite tal cosa.

Por otro lado, permitiendo *defensa en equipo* implica múltiples declaraciones que respaldan una declaración puede combinarse para defenderse contra declaraciones desafiantes.

Las funciones de etiquetado son lo suficientemente flexibles como para modelar las intuiciones mencionadas. En [19], Hecham et al. presentaron cuatro funciones de etiquetado diferentes para caracterizarlos a todos. Estas funciones de etiquetado asignan etiquetas *unsup* de la misma manera pero difieren en la forma en que asignan *in*, *out* y *ambig*. Concretamente, tendremos los siguientes etiquetados:

(a) L^{TD}_{block}: etiquetado con defensa en equipo, con bloqueo de ambigüedad.

(b) L^{TD}_{prop}: etiquetado con defensa en equipo, con propagación de la ambigüedad.

(c) L^{noTD}_{block}: etiquetado sin defensa en equipo, con bloqueo de ambigüedad.

(d) L^{noTD}_{prop}: etiquetado sin defensa en equipo, con propagación de la ambigüedad.

Una declaración está etiquetada *in* si y solo si tiene un soporte completo rebatiblemente aceptado (i.e. hay una derivación rebatiblemente aceptada para cada una de sus premisas), Sus declaraciones de apoyo son superiores a sus socavaciones **de acuerdo con el etiquetado i.e. defensa en equipo y manejo de ambigüedades** y la regla de declaración es superior a cualquier regla aceptada que la ataque.

1. *a)* $L^{TD}_{block}(s) = in$ *si y solo si* s *tiene un soporte completo "in" y* $\forall s' \in \mathcal{V}_A(s)$ *que ataca mediante un undercut* s *en una premisa* $f \in Premise(s)$ *tal que* $L^{TD}_{block}(s') = in$, $\exists s'' \in \mathcal{V}_S(s)$ *que soporta* f *tal que* $L^{TD}_{block}(s'') = in$ *o* ($L^{TD}_{block}(s'') = in$ *y* s'' *es superior a* s'*).*

   b) $L^{TD}_{prop}(s) = in$ *si y solo si* s *tiene un soporte completo "in" y* $\forall s' \in \mathcal{V}_A(s)$ *que ataca mediante un undercut* s *en una premisa* $f \in Premise(s)$ *tal que* $L^{TD}_{prop}(s') \in \{in, ambig\}$, $\exists s'' \in \mathcal{V}_S(s)$ *que soporta* f *tal que* $L^{TD}_{prop}(s'') = in$ *o* ($L^{TD}_{prop}(s'') = in$ *y* s'' *es superior a* s'*).*

   c) $L^{noTD}_{block}(s) = in$ *si y solo si* s *tiene un soporte completo "in" y* $\forall f \in Premise(s)$, $\exists s' \in \mathcal{V}_S(s)$ *que soporta* f *y* $L^{noTD}_{block}(s') = in$ *o* ($L^{noTD}_{block}(s') = in$ *y* $\forall s'' \in \mathcal{V}_A(s)$ *que ataca mediante un undercut* s *on* f *tal que* $L^{noTD}_{block}(s'') = in$, s' *es superior a* s''*).*

   d) $L^{noTD}_{prop}(s) = in$ *si y solo si* s *tiene un soporte completo "in" y* $\forall f \in Premise(s)$, $\exists s' \in \mathcal{V}_S(s)$ *que soporta* f *y* $L^{noTD}_{prop}(s') = in$ *o* ($L^{noTD}_{prop}(s') = in$ *y* $\forall s'' \in \mathcal{V}_A(s)$ *que ataca mediante un undercut* s *on* f *tal que* $L^{noTD}_{prop}(s'') \in \{in, ambig\}$, s' *es superior a* s''*).*

Una declaración está etiquetada *out* si y solo si tiene un soporte completo aceptado y existe un ataque rebatiblemente aceptado que es superior a su soporte **de acuerdo con el etiquetado**.

2. *a)* $L^{TD}_{block}(s) = out$ *si y solo si* s *tiene un soporte completo "in" y* $\exists s' \in \mathcal{V}_A(s)$ *que ataca mediante un undercut* s *en una premisa* $f \in Premise(s)$ *tal que* $L^{TD}_{block}(s') = in$ *y* $\forall s'' \in \mathcal{V}_S(s)$ *que soporta* f, $L^{TD}_{block}(s'') \neq in$ *y* ($L^{TD}_{block}(s'') \neq in$ *o* s' *es superior a* s''*).*

b) $L^{TD}_{prop}(s) = out$ *si y solo si* s *tiene un soporte completo "in" y* $\exists s' \in \mathcal{V}_A(s)$ *que ataca mediante un undercut* s *en una premisa* $f \in Premise(s)$ *tal que* $L^{TD}_{prop}(s') \in \{in, ambig\}$ $y \forall s'' \in \mathcal{V}_S(s)$ *que soporta* f, $L^{TD}_{prop}(s'') \neq in$ y $(L^{TD}_{prop}(s'') \neq in$ o s' *es superior a* $s'')$.

c) $L^{noTD}_{block}(s) = out$ *si y solo si* s *tiene un soporte completo "in" y* $\exists f \in Premise(s)$ *tal que* $\forall s' \in \mathcal{V}_S(s), L^{noTD}_{block}(s') \neq in$ y *if* $L^{noTD}_{block}(s') = in$, *entonces* $\exists s'' \in \mathcal{V}_A(s)$ *tal que hay un undercut* f, $L^{noTD}_{block}(s'') = in$ y s'' *es superior a* s'.

d) $L^{noTD}_{prop}(s) = out$ *si y solo si* s *tiene un soporte completo "in" y* $\exists f \in Premise(s)$ *tal que* $\forall s' \in \mathcal{V}_S(s), L^{noTD}_{prop}(s') \neq in$ y *if* $L^{noTD}_{prop}(s') = in$, *entonces* $\exists s'' \in \mathcal{V}_A(s)$ *tal que hay un undercut* f, $L^{noTD}_{prop}(s'') \in \{in, ambig\}$ y s'' *es superior a* s'.

Una declaración está etiquetada *ambig* si no es aceptada ni rechazada y tiene un soporte completo aceptado o ambiguo.

3. $L(s) = ambig$ *si y solo si* $L(s) \notin \{in, out\}$ y s *tiene un* in, *o* $ambig$ *soporte completo.*

Una declaración está etiquetada *unsup* si tiene una premisa que no es soportada por una declaración aceptada o ambigua.

4. $L(s) = unsup$ *si y solo si* $L(s) \neq out$ y $\exists f \in Premise(s)$ *tal que* $\nexists s' \in \mathcal{V}_S(s)$ *soporte* f *tal que* $L(s') \in \{in, ambig\}$.

Ejemplo 2.2.32. En nuestro ejemplo corriente, L^{prop}_{TD} daría el grafo etiquetado de la figura 2.9 considerando el conjunto de preferencias dado por $\prec$ tal que:

- $sky(clear) \rightarrow weather(nice) \equiv forecast(rain) \rightarrow weather(rain)$.
- $tiring(bicycle) \rightarrow notTake(bicycle) \succ free(bicycle) \rightarrow take(bicycle)$.
- $tiring(bicycle) \rightarrow notTake(bicycle) \succ fast(bicycle) \rightarrow take(bicycle)$.

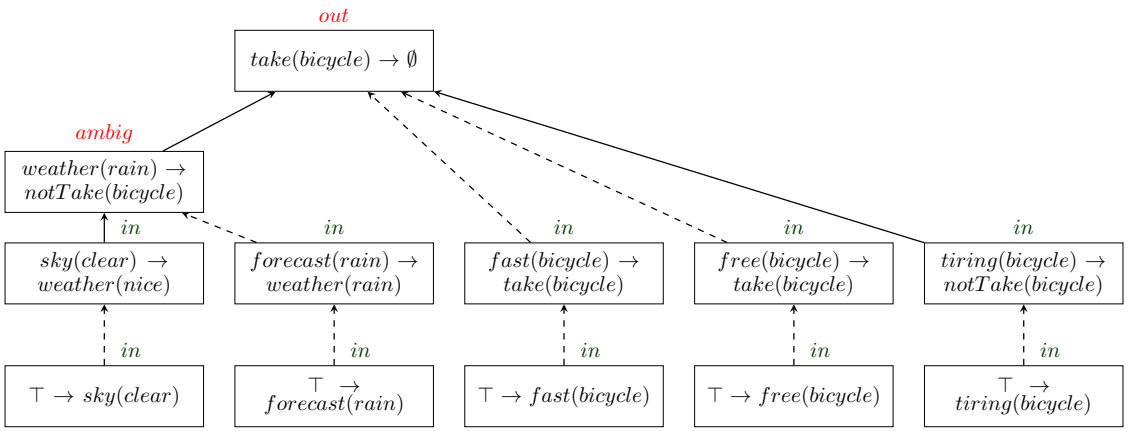

Fig. 2.9: $\mathcal{SG}^{L^{prop}_{TD}}_{\mathcal{KB}}$ del ejemplo 2.2.18.

Hecham et al. [19] demostró cómo las funciones de etiquetado se relacionan con la existencia de una prueba para átomos con respecto a el razonamiento rebatible según lo descrito por diferentes expertos en lógica rebatible [10]. En lógicas rebatibles, los átomos se evalúan usando pruebas, que son secuencias de átomos marcados que describen entailment.

Finalmente, las siguientes son las complejidades de construir y etiquetar un grafo de declaraciones.

Proposición 2.2.33. ***Complejidad de construcción y etiquetado de un grafo de declaraciones [18].*** *La construcción del grafo de declaraciones y calcular sus etiquetas tiene "combined time complexity" (la entrada contiene el conjunto de reglas, hechos y la query):* 2ExpTime-complete *y "data time complexity"* PTime-complete *(la entrada contiene solo el conjunto de hechos mientras que las reglas y la query se supone que son fijas) para el lenguaje* $\mathcal{L}$.

3. SEMÁNTICAS DE RANKING PARA GRAFOS DECLARATIVOS

Como se discutió en las secciones anteriores, las semánticas de ranking permiten evaluar la fuerza de los argumentos y compararlos con respecto a otros. Muchas de estas semánticas se han definido en la literatura, ambos para sistemas de argumentación clásicos [1] como para sistemas de argumentación bipolares [2] (con ejes de soporte y ataque).

En este trabajo, presentaremos un esquema para generar rankings en el contexto de grafos de declaraciones.

Las semánticas de ranking no deben confundirse con las funciones de etiquetado. En contraste, las semánticas de ranking permitir evaluar la fuerza de las declaraciones desde una perspectiva diferente que es alternativa a los valores de verdad asignados por las funciones de etiquetado definidas anteriormente.

3.1. Nociones básicas

Definimos una semántica de ranking para grafos de declaraciones como:

Definición 3.1.1. Semántica de ranking. Dado un grafo de declaraciones $\mathcal{SG} = (\mathcal{V}, \mathcal{E}_S, \mathcal{E}_A)$, una semántica de ranking $\leq$: $\mathcal{SG} \mapsto (\mathcal{V} \times \mathcal{V})$ toma un grafo de declaraciones y produce un ranking entre sus declaraciones $\leq_{\mathcal{SG}} \subseteq \mathcal{V} \times \mathcal{V}$ (i.e. una relación binaria que es total y transitiva). Denotaremos $s_1 <_{\mathcal{SG}} s_2$ si y solo si $s_1 \leq_{\mathcal{SG}} s_2$ y $s_2 \not\leq_{\mathcal{SG}} s_1$. También, $s_1 \equiv_{\mathcal{SG}} s_2$ si y solo si $s_1 \leq_{\mathcal{SG}} s_2$ y $s_2 \leq_{\mathcal{SG}} s_1$.

Una forma de producir tal orden es usando una función de peso. Las funciones de peso ya se han estudiado en la literatura con éxito, como hemos visto anteriormente. Estas funciones asignan un número real en el intervalo unitario $[0, 1]$ para todos y cada uno de los elementos en el grafo. Además, debido a esto, la relación de orden que producen es total: se puede comparar cada par de elementos.

Definición 3.1.2. Función de peso. Dado un grafo de declaraciones $\mathcal{SG} = (\mathcal{V}, \mathcal{E}_S, \mathcal{E}_A)$, una función de peso $W : \mathcal{SG} \mapsto (\mathcal{V} \mapsto [0, 1])$ toma un grafo de declaraciones y produce un peso para cada declaración $W_{\mathcal{SG}} : \mathcal{V} \mapsto [0, 1]$. Por otra parte, esta función de peso induce una semántica de ranking $\leq^W_{\mathcal{SG}} \subseteq \mathcal{V} \times \mathcal{V}$ tal que $s_1 \leq^W_{\mathcal{SG}} s_2$ si y solo si $W_{\mathcal{SG}}(s_1) \leq W_{\mathcal{SG}}(s_2)$.

De manera de definir fácilmente funciones de peso para obtener rankings entre las declaraciones, definiremos el concepto de acumulador. Un acumulador es simplemente una función que, dado un conjunto de declaraciones positivo (de soporte) y otro negativo (de ataque) con ciertos pesos, devuelve un peso combinado.

Definición 3.1.3. Función de acumulación. Dados dos conjuntos de declaraciones $S^+, S^- \subseteq S$ y una función de peso $W : S \mapsto [0, 1]$, un acumulador es una función $F_W : S \times S \mapsto [0, 1]$ tal que:

- Sea $S' \subseteq S^+$, $F_W(S', S^-) \leq F_W(S^+, S^-)$ i.e. aumentar el primer conjunto produce un peso mayor.
- Sea $S' \subseteq S^-$, $F_W(S^+, S') \geq F_W(S^+, S^-)$ i.e. disminuir el segundo conjunto produce un peso mayor.

- Sea $s \in S^+$ y $S' = S^+ \cup \{s'\} - \{s\}$ tal que $W(s') \leq W(s)$, $F_W(S', S^-) \leq F_W(S, S^-)$ i.e. mejorar el primer conjunto produce un peso mayor.
- Sea $s \in S^-$ y $S' = S^- \cup \{s'\} - \{s\}$ tal que $W(s') \leq W(s)$, $F_W(S^+, S') \geq F_W(S^+, S)$ i.e. empeorar el segundo conjunto produce un peso mayor.

Partiendo de la función utilizada para la semántica de ranking categoriser, una de las primeras para argumentación abstracta, podemos capturar su forma de acumular la fuerza de un conjunto de la siguiente manera.

Definición 3.1.4. Acumulador categoriser. Dados los conjuntos de declaraciones S^+, S^- y una función de peso $W : S \mapsto [0, 1]$, el acumulador categoriser es la función: $Cat_W(S^+, S^-) = (1 - \frac{1}{1+\sum\limits_{s \in S^+} W(s)})(\frac{1}{1+\sum\limits_{s \in S^-} W(s)})$.

Adicionalmente, proponemos utilizar también la función logística, una función bien conocida y estudiada en la literatura, con distintos usos en varios campos de la inteligencia artificial.

Definición 3.1.5. Acumulador logístico. Dados los conjuntos de declaraciones S^+, S^- y una función de peso $W : S \mapsto [0, 1]$, el acumulador logístico es la función: $\sigma_W(S^+, S^-) = \frac{1}{1+e^E}$ donde $E = \sum\limits_{s \in S^-} W(s) - \sum\limits_{s \in S^+} W(s)$.

Es simple ver que ambas funciones son acumuladores. Además, son continuas, derivables y mapean los conjuntos positivo S^+ y negativo S^- al intervalo unitario $[0, 1]$. La principal diferencia es que cuando el conjunto positivo es vacío ($S^+ = \emptyset$) el acumulador categoriser devolverá $Cat_W(\emptyset, S^-) = 0$, mientras que en el logístico $\sigma(\emptyset, S^-) \neq 0$. De hecho, $\sigma(\emptyset, \emptyset) =$ 0,5, en otras palabras, ante la falta de pruebas, el primer acumulador decide no creer en absoluto, mientras que el segundo devuelve un valor de fuerza medio.

3.2. Principios para semánticas de ranking

A continuación, definamos algunas propiedades básicas para comprender mejor la subyacente mecánica de las semánticas de ranking en el contexto de grafos de declaraciones. Con ese fin, tendremos en cuenta los principios bien conocidos de semánticas de ranking previamente definidos en la literatura [11, 2].

Primero, definamos algunas nociones que nos serán de utilidad durante esta sección.

La unión de dos grafos de declaraciones es el resultado de unir sus declaraciones y agregar los ejes de soporte y ataque necesarios.

Definición 3.2.1. Unión. Dados $\mathcal{SG} = (\mathcal{V}, \mathcal{E}_S, \mathcal{E}_A)$, $\mathcal{SG}' = (\mathcal{V}', \mathcal{E}'_S, \mathcal{E}'_A)$ grafos de declaraciones con restricciones negativas $\mathcal{N}_1$ y $\mathcal{N}_2$, decimos que la unión es $\mathcal{SG} \cup \mathcal{SG}' = (\mathcal{V} \cup \mathcal{V}', \mathcal{E}''_S, \mathcal{E}''_A)$ donde dadas dos declaraciones $s_1, s_2 \in (\mathcal{V} \cup \mathcal{V}')$:

- $(s_1, s_2) \in \mathcal{E}''_S$ si y solo si $Conc(s_1) \neq \emptyset$ y $Conc(s_1) \in Premise(s_2)$.
- $(s_1, s_2) \in \mathcal{E}''_A$ si y solo si $\exists f \in Premise(s_2)$ tal que f y $Conc(s_1)$ están en conflicto con respecto a $\mathcal{N}_1 \cup \mathcal{N}_2$.

A continuación definiremos algunas propiedades para caracterizar nuestras semánticas de ranking.

Definición 3.2.2. Dado un grafo de declaraciones $\mathcal{SG} = (\mathcal{V}, \mathcal{E}_S, \mathcal{E}_A)$ y una semántica de ranking $\leq$: $\mathcal{SG} \mapsto (\mathcal{V} \times \mathcal{V})$, decimos que satisface:

- **Soporte (Sop):** si cuando $s \in \mathcal{V}$ no tiene un soporte para cierta premisa, entonces para todo $s' \in \mathcal{V}$, $s \leq_{\mathcal{SG}} s'$.
- **Terminación acíclica (TA):** si cuando no hay ciclos en el grafo de declaraciones, entonces se pueden calcular todos los pesos.
- **Direccionalidad (Dir):** si la fuerza de una declaración solo depende de las declaraciones que están conectadas a ella. Dado un grafo de declaraciones $\mathcal{SG}' = (\mathcal{V} \cup \{s_1\}, \mathcal{E}'_S, \mathcal{E}'_A)$, entonces $\forall s_2, s_3 \in \mathcal{V}$ tal que no hay camino dirigida (ya sea con soportes o ataques) de s_1 hacia s_2 o s_3 en $\mathcal{SG}'$, entonces $s_2 \leq_{\mathcal{SG}} s_3$ si y solo si $s_2 \leq_{\mathcal{SG}'} s_3$.
- **Precedencia de especificidad disminuida (PED):** si ser menos específico produce una mayor fuerza para una declaración. Dadas dos declaraciones completamente soportadas $s_1, s_2 \in \mathcal{V}$, que no contienen reglas hecho (sin premisa vacía), donde $Premise(s_2) = Premise(s_1) \cup \{p\}$ donde p no tiene ataques, solo soportes, entonces $s_1 \geq_{\mathcal{SG}} s_2$.
- **Precedencia de especificidad aumentada (PEA):** si ser más específico produce una mayor fuerza para una declaración. Dadas dos declaraciones completamente soportadas $s_1, s_2 \in \mathcal{V}$, que no contienen reglas hecho (sin premisa vacía),donde $Premise(s_2) = Premise(s_1) \cup \{p\}$ donde p no tiene ataques, solo soportes, entonces $s_2 \geq_{\mathcal{SG}} s_1$.

3.3. Semánticas de ranking por acumulación intra-premisa

En nuestros grafos de declaraciones, los nodos contienen declaraciones de la forma $s = (p_1 \wedge ... \wedge p_n \rightarrow c)$ con $p_1, ..., p_n, c$ átomos. En otras palabras, las declaraciones representan pasos de inferencia. Desde un punto de vista lógico, la validez de los pasos de inferencia siempre dependen de la validez de sus premisas. Además de eso, las declaraciones pueden atacarse o apoyarse mutuamente.

Teniendo en cuenta esas nociones, definiremos un mecanismo para producir pesos para cada declaración cuantificando la fuerza de cada una de sus premisas $p_1, ..., p_n$ por separado (**intra-premisa**) y luego lo extendemos para producir un valor de fuerza para la declaración en sí.

Definición 3.3.1. Ranking por acumulación intra-premisa. Sea $\mathcal{SG} = (\mathcal{V}, \mathcal{E}_S, \mathcal{E}_A)$ un grafo de declaraciones, la declaración $s = (p_1 \wedge ... \wedge p_n \rightarrow c) \in \mathcal{V}$ y F una función acumuladora, definimos la fuerza de s como:

$$
\begin{aligned}
W(s) =& 1 \text{ si } Premise(s) = \{\top\}, \text{ en caso contrario,} \\
W(s) =& \prod_{i=1,...,n} W_F^{\mathcal{SG}}(p_i), \text{ donde} \\
W(p_i) =& F_W(\{s' : s' \in \mathcal{V} \text{ soporta } p_i\}, \{s' : s' \in \mathcal{V} \text{ ataca } p_i\})
\end{aligned}
$$

En este caso, decimos que W produce un **ranking por acumulación intra-premisa**, donde dadas dos declaraciones $s, s' \in \mathcal{V}$, $s \leq_W s'$ si y solo si $W(s) \leq W(s')$.

Proposición 3.3.2. *Los ranking por acumulación intra-premisa satisfacen Sop, TA, Dir y PED.*

Demostración. Veamos que las propiedades se satisfacen:

- Sop: se desprende de la definición.
- TA: se desprende de que los pesos de una declaración se determinan con respecto a los hijos de la declaración.
- Dir: se desprende de que los pesos de una declaración se determinan con respecto a los hijos de la declaración, por lo tanto, como ninguno de s_2, s_3 ni sus hijos cambia su peso debido a la declaración adicional s_1, $s_2 \leq_{\mathcal{SG}} s_3$ si y solo si $s_2 \leq_{\mathcal{SG}'} s_3$.
- PED: dadas dos declaraciones totalmente soportadas $s_1, s_2 \in \mathcal{V}$ sin premisa vacía, donde $Premise(s_2) = Premise(s_1) \cup \{p\}$, entonces $W_F^{\mathcal{SG}}(s_2) = W_F^{\mathcal{SG}}(s_1) * W_F^{\mathcal{SG}}(p)$, donde $W_F^{\mathcal{SG}}(p) \in [0, 1]$.

  Por lo tanto, $W_F^{\mathcal{SG}}(s_2) \leq W_F^{\mathcal{SG}}(s_1)$.

□

Ejemplo 3.3.3. Considere los siguientes grafos de declaraciones con declaraciones y ejes como se muestra en sus respectivos gráficos a continuación, con valores de peso W marcados en los nodos:

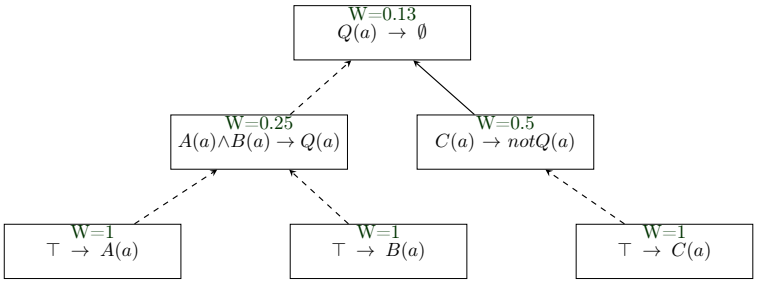

Fig. 3.1: Ejemplo (a) de acumulador Cat.

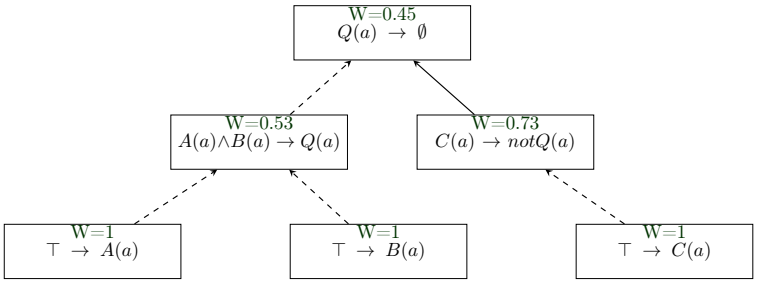

Fig. 3.2: Ejemplo (b) de acumulador σ.

3.4. Semánticas de ranking por acumulación inter-premisa

Ahora, definiremos un mecanismo para producir pesos para cada declaración cuantificando la fuerza de sus premisas $p_1, ..., p_n$ de forma combinada (**inter-premisa**) y luego lo extenderemos para producir un valor de fuerza para la declaración en sí.

Definición 3.4.1. Ranking por acumulación inter-premisa. Sea $\mathcal{SG} = (\mathcal{V}, \mathcal{E}_S, \mathcal{E}_A)$ un grafo de declaraciones, la declaración $s = (p_1 \wedge ... \wedge p_n \rightarrow c) \in \mathcal{V}$ y F una función acumuladora, definimos la fuerza de s como:

$$W(s) = 1 \text{ si } Premise(s) = \{\top\}, \text{ en caso contrario,}$$

$$W(s) = \begin{cases} 0 & \text{si } \exists p_i \text{ sin soporte} \\ F_W(\{s' : s' \in \mathcal{V} \text{ soporta } s\}, \{s' : s' \in \mathcal{V} \text{ ataca } s\}) & \text{en caso contrario} \end{cases}$$

En este caso, decimos que W produce un **ranking por acumulación inter-premisa**, donde dadas dos declaraciones $s, s' \in \mathcal{V}$, $s \leq_W s'$ si y solo si $W(s) \leq W(s')$.

Proposición 3.4.2. *Los ranking por acumulación intra-premisa satisfacen Sop, TA, Dir y PEA.*

Demostración. Veamos que las propiedades se satisfacen:

- Sop: se desprende de la definición.
- TA: se desprende de que los pesos de una declaración se determinan con respecto a los hijos de la declaración.
- Dir: se desprende de que los pesos de una declaración se determinan con respecto a los hijos de la declaración, por lo tanto, como ninguno de s_2, s_3 ni sus hijos cambia su peso debido a la declaración adicional s_1, $s_2 \leq_{\mathcal{SG}} s_3$ si y solo si $s_2 \leq_{\mathcal{SG}'} s_3$.
- PEA: dadas dos declaraciones totalmente soportadas $s_1, s_2 \in \mathcal{V}$ sin premisa vacía, donde $Premise(s_2) = Premise(s_1) \cup \{p\}$, entonces $W_F^{\mathcal{SG}}(s_1) = F_W(\{s' : s' \in \mathcal{V} \text{ soporta } s_1\}, \{s' : s' \in \mathcal{V} \text{ ataca } s_1\})$, mientras que $W_F^{\mathcal{SG}}(s_2) = F_W(\{s' : s' \in \mathcal{V} \text{ soporta } s_1\} \cup \{s' : s' \in \mathcal{V} \text{ soporta } p\}, \{s' : s' \in \mathcal{V} \text{ ataca } s_1\})$ por la definición de s_1 y s_2, luego por la definición de acumulador (creciente en el primer conjunto), $W_F^{\mathcal{SG}}(s_2) \geq W_F^{\mathcal{SG}}(s_1)$.

□

Ejemplo 3.4.3. Considere los siguientes grafos de declaraciones con declaraciones y ejes como se muestra en sus respectivos gráficos a continuación, con valores de peso W marcados en los nodos:

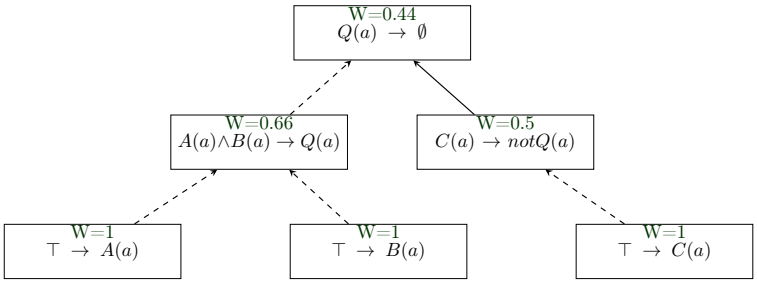

Fig. 3.3: Ejemplo (a) inter-premisa con acumulador Cat.

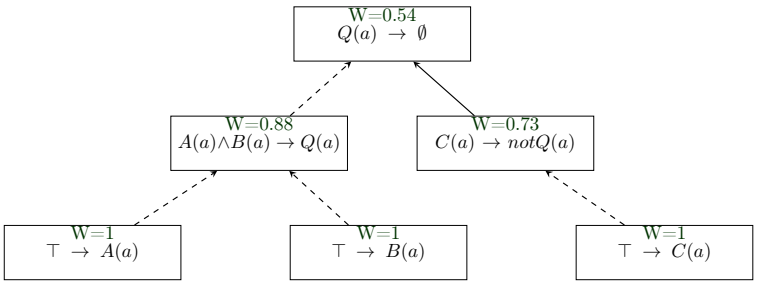

Fig. 3.4: Ejemplo (b) inter-premisa con acumulador σ.

3.5. Semánticas de conjuntos utilizando rankings

Como vimos anteriormente, mediante preferencias, el razonamiento rebatible es capaz de definir aceptabilidad mediante etiquetados tal como para los sistemas de argumentación. En particular, lo hace de manera de capturar las distintas intuiciones de la lógica rebatible. De esta forma, es posible razonar aun cuando hay posibles derivaciones contradictorias.

Como vimos, un etiquetado aplicado a un grafo de declaraciones $\mathcal{SG} = (\mathcal{V}, \mathcal{E}_S, \mathcal{E}_A)$ y una relación de preferencia $\succ$ es una función $L : \mathcal{V} \rightarrow \{in, out, ambig, unsup\}$ que toma como entrada una declaración $s \in \mathcal{V}$ y devuelve una etiqueta.

La relación de preferencia puede ser tomada como parte de la entrada, pero también puede ser generada mediante las semánticas de ranking que definimos. Es decir, podemos generar rankings mediante nuestras funciones de acumulación sin suponer nada sobre las reglas, solo con la estructura del grafo, y utilizar ese ranking como nuestra relación de preferencias para razonar rebatiblemente y obtener etiquetas.

Ejemplo 3.5.1. Continuando con el ejemplo 2.2.18, obtenemos los siguientes pesos, denotados por W en los nodos, y las siguientes etiquetas según $L = L^{prop}_{TD}$. Notar que ahora la query tiene etiqueta *ambig* ya que tiene un ataque de la declaración ($tiring(bicycle \rightarrow notTake(bicycle)$ con etiqueta *in* al cual ningún soporte es superior.

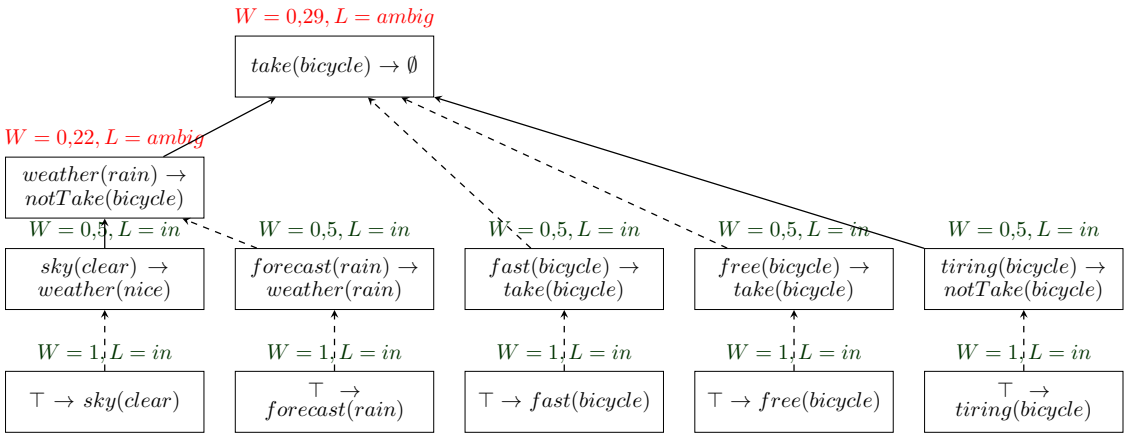

Fig. 3.5: Ejemplo etiquetado usando ranking inter-premisa mediante acumulador Cat.

4. EXPLICACIONES

Como discutimos en la introducción, el propósito de esta tesis es modelar un marco explicativo para aclarar el proceso de razonamiento con el cual se obtienen conclusiones de tal manera que un usuario humano no se vea abrumado con información innecesaria.

Por lo tanto, el objetivo de dicho marco es proporcionar un procedimiento gradual de aumento de conocimiento que permita al usuario desarrollar los pasos de inferencia considerados poco claros al priorizar los aspectos clave de dicho proceso de razonamiento.

4.1. Explicaciones en sistemas de argumentación

Inicialmente, definiremos nuestro procedimiento de aumento de conocimiento para los sistemas de argumentación clásicos [15].

Recordemos el concepto de subgrafos (definition 2.1.4), que son simplemente los subgrafos de un sistema de argumentación.

Sea AF un sistema de argumentación, una expansión toma cualquiera de sus subgrafos y agrega un subconjunto de argumentos de AF junto con todos los ataques presentes en AF con respecto al conjunto actualizado de argumentos. En otras palabras, produce otro subgrafo.

Definición 4.1.1. Expansión. Dado un sistema de argumentación $AF = (A, R)$ y un subgrafo $AF' = (A', R')$, decimos que:

$$AF' \xrightarrow{S} AF''$$

es una expansión de AF' con respecto a AF si y solo si $AF'' = (S \cup A', R'')$ es un subgrafo de AF y $S \cup A' \subseteq A$.

Ejemplo 4.1.2. Dado el sistema de argumentación $AF = (\{a, b, c, d, e\}, \{(a, b), (b, c), (a, c), (c, d), (e, b)\})$ como en la figura 4.1 y subgrafos $AF' = (\{a, b\}, \{(a, b)\})$, $AF'' = (\{a, b, c\}, \{(a, b), (b, c), (a, c)\})$, decimos que AF'' es el resultado de una expansión de AF' con respecto a AF y lo denotamos $AF' \xrightarrow{\{c\}} AF''$.

Luego, una expansión de argumento toma un sistema de argumentación AF, un subgrafo AF' y una discusión v de AF' y expande el subgrafo agregando un subconjunto de argumentos conectados a v con respecto a AF.

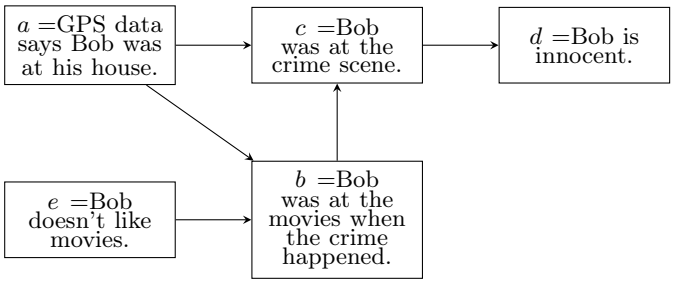

Fig. 4.1: Sistema de argumentación debatiendo si *Bob is innocent.*

Definición 4.1.3. Expansión de argumentos. Dado un sistema de argumentación $AF = (A, R)$, decimos que:

$$f : \mathcal{AF} \times \mathcal{AF} \times 2^{Arg} \mapsto 2^{Arg}$$

es una función de expansión de argumento si y solo si se le dado un subgrafo $AF' = (A', R')$ de AF y argumento $v \in A'$, $AF' \xrightarrow{f(AF,AF',v)} AF''$ es una expansión y se sostiene que:

- **(Inclusión)** $f(AF, AF', v) \subseteq A$.
- **(Conectado)** $\forall v' \in f(AF, AF', v)$, existe un camino desde v' a v en AF''.
- **(Completo)** $\forall v' \in A$ tal que existe un camino desde v' a v en AF, hay un número finito de pasos $n \in \mathbb{N}$ tal que $AF' = AF^0 \xrightarrow{f(AF,AF^0,v)} AF^1 \xrightarrow{f(AF,AF^1,v)} ... \xrightarrow{f(AF,AF^{n-1},v)} AF^n = (A^n, R^n)$ and $v' \in A^n$.

Llamaremos a esto una expansión del argumento y (abusivamente) lo denotaremos como:

$$AF' \xrightarrow{(f,v)} AF''$$

Ejemplo 4.1.4. Siguiendo el ejemplo anterior, dada una función de expansión de argumento f tal que $f(AF, AF', a) = \{c\}$, decimos que $AF' \xrightarrow{(f,a)} AF''$ es una expansión de argumento.

Finalmente, juntemos todos los conceptos previamente definidos para definir lo que llamaremos una explicación. Como discutimos anteriormente, una explicación es un procedimiento que proporciona un aumento gradual del conocimiento. Por lo tanto, proponemos modelar explicaciones como una sucesión de pasos de expansión. Además, no será necesario ampliar nuestro conocimiento hasta que se revele por completo, siempre y cuando se llegue a la conclusión original.

Definición 4.1.5. Explicación. Dado un sistema de argumentación $AF = (A, R)$, una función de etiquetado L, una función de expansión de argumento f y un argumento v, una explicación para v con respecto a AF y f es una sucesión de expansiones:

$$AF^0 \xrightarrow{(f,v_0)} AF^1 \xrightarrow{(f,v_1)} ... \xrightarrow{(f,v_{n\text{-}1})} AF^n$$

donde $AF^0 = (\{v\}, \emptyset, \emptyset)$, $\forall i$, v_i es un argumento de AF^i y $L_{AF}(v) = L_{AF^n}(v)$.

Ejemplo 4.1.6. Volviendo al ejemplo de la figura 4.1, una posible explicación para el argumento $d =$*Bob is innocent* sería el representado en la figura 4.2 si ampliamos nuestro conocimiento usando la función de expansión de argumentos f_1 donde $f_1(AF, AF^0, d) = \{b, c\}$ y $f_1(AF, AF^1, d) = \{a, e\}$. Alternativamente, la figura 4.3 refleja una explicación usando f_2 donde $f_2(AF, AF^0, d) = \{a, c\}$ y $f_2(AF, AF^1, d) = \{b, e\}$.

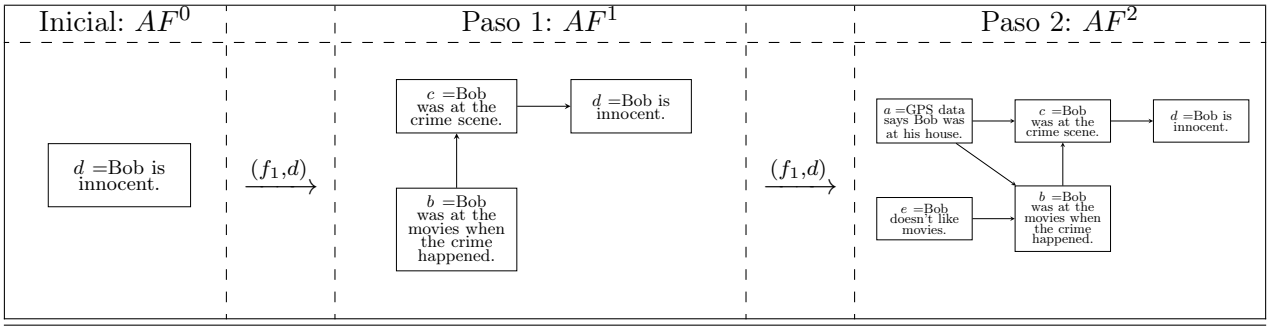

Fig. 4.2: Explicación (a) para *Bob is innocent*

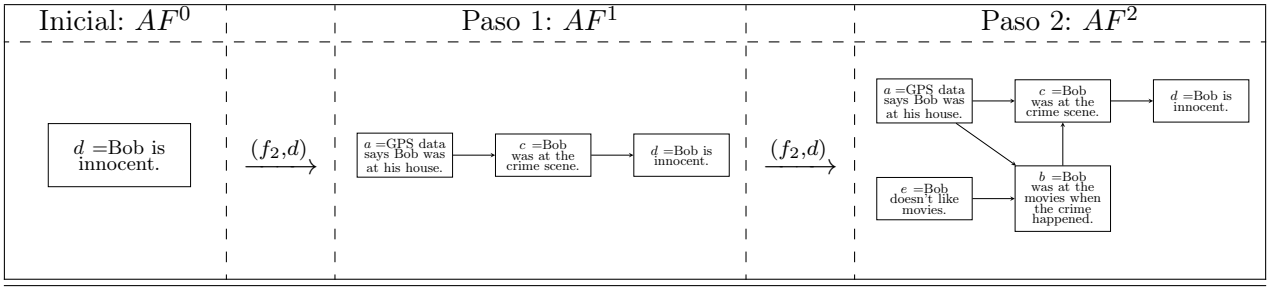

Fig. 4.3: Explicación (b) para *Bob is innocent*

4.2. Esquemas de expansión

En esta sección definiremos algunos esquemas de expansión útiles y mostraremos los diferentes comportamientos que tienen.

Tenga en cuenta que los marcos de argumentación considerados en esta sección tendrán un conjunto de ataques **finito**. Esta es una clave condición debido al hecho de que la mayoría de los esquemas de expansión no podrían satisfacer la textbf condición conectada (en la definición de expansiones de argumentos) como marcos de argumentación infinitos requeriría una cantidad infinita de pasos de expansión para llegar a ciertos argumentos.

Primero definamos la noción de distancia entre argumentos en un marco.

Definición 4.2.1. Distancia. Dado un sistema de argumentación $AF = (A, R)$ y dos argumentos $v, v' \in A$ de tal manera que haya un camino de ataques entre v y v' en AF, la distancia entre v y v' es:

$$d_{AF}(v, v') = \text{mín}\,\#(\{P \mid P \text{ es un camino de ataques desde } v \text{ a } v' \text{ en } AF\})$$

Si no hay un camino de ataques entre v y v' en AF decimos que la distancia es infinita.

Dado un sistema de argumentación $AF = (A, R)$, definamos los siguientes esquemas de expansión.

Una expansión "breadth first" agrega un conjunto de atacantes no especificados del sistema original que están más cerca del argumento expandido.

Definición 4.2.2. Expansión breadth first. Una función de expansión de argumentos f_{BF} es llamada breadth first si y solo si dado cualquier subgrafo $AF' = (A', R')$ y argumento $v \in A'$:

$$\forall v' \in f_{BF}(AF, AF', v),\ \forall v'' \in (A - A')\ d_{AF}(v', v) \leq d_{AF}(v'', v)$$

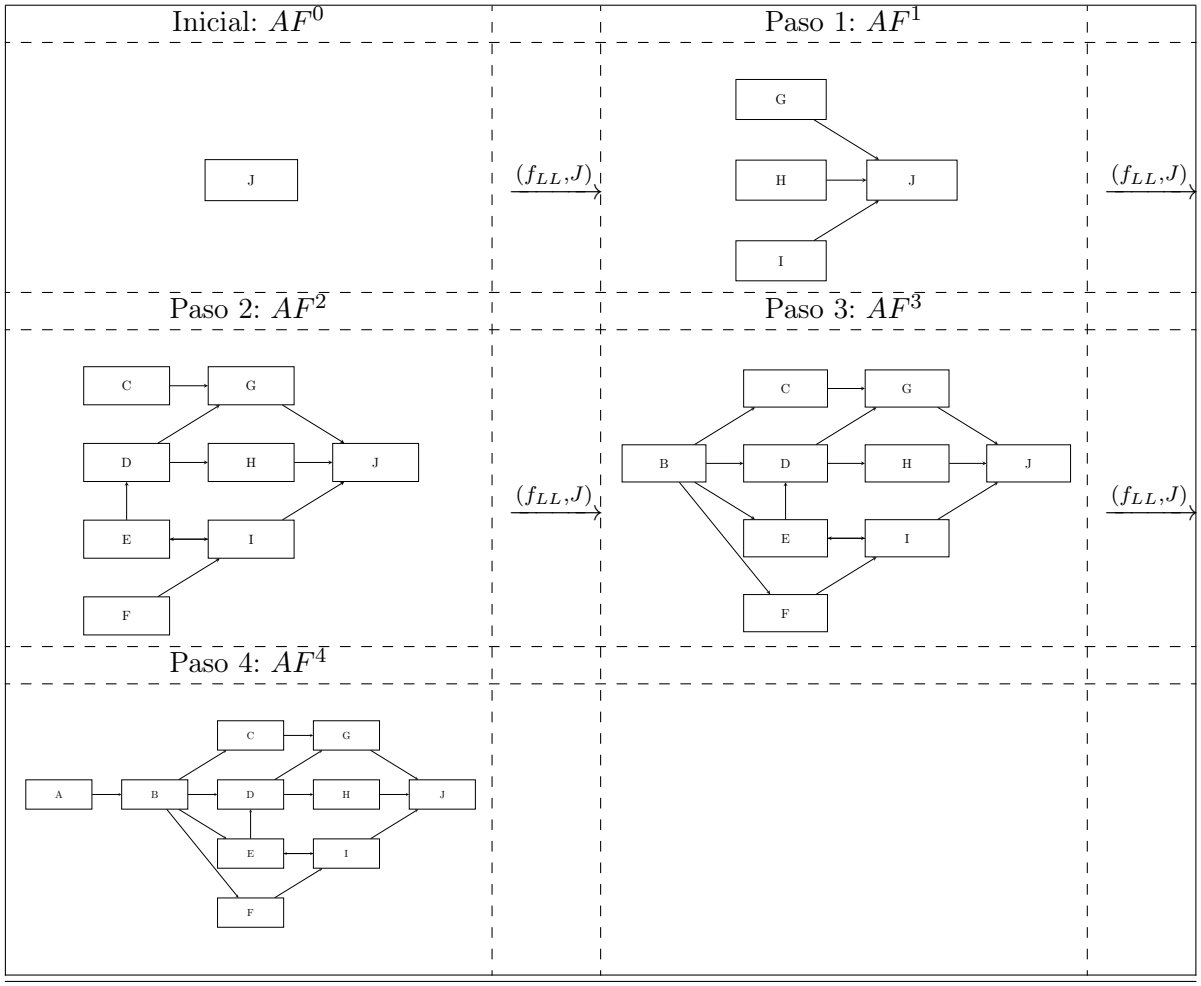

Fig. 4.4: Explicación nivel por nivel

Una expansión de nivel por nivel agrega todos los argumentos más cercanos al argumento expandido.

Definición 4.2.3. Expansión nivel por nivel. Una función de expansión de argumentos f_{LL} es llamada nivel por nivel si y solo si dado cualquier subgrafo $AF' = (A', R')$ y argumento $v \in A'$:

$$f_{LL}(AF, AF', v) = \{v' \in (A - A') \mid \forall v'' \in (A - A') \\ d_{AF}(v', v) \leq d_{AF}(v'', v)\}$$

Proposición 4.2.4. *Las expansiones de nivel por nivel son breadth first, debido al hecho de que todos los Los argumentos son tan cercanos como cualquier otro al argumento expandido.*

Ejemplo 4.2.5. Describamos el comportamiento de las expansiones de nivel por nivel con el siguiente ejemplo en la figura 4.4.

A continuación, reutilizaremos los rankings $\leq$ definido en la sección 2.1.3 para generar expansión esquemas que tienen en cuenta la fuerza de los argumentos.

Dado un ranking $\leq$ sobre todos los argumentos de AF, definimos el siguiente esquema de expansiones.

Definición 4.2.6. Eexpansión breadth first ranked. Una función de expansión de argumentos $f_{BF,\leq}$ es llamada expansión breadth first ranked si y solo si dado cualquier subgrafo $AF' = (A', R')$ y argumento $v \in A'$:

$$f_{BF,\leq}(AF, AF', v) = \{v' \mid \forall v'' \in (A - A') \\ d_{AF}(v, v') < d_{AF}(v, v'') \text{ o } d_{AF}(v, v') \equiv d_{AF}(v, v'') \wedge v' \geq v''\}$$

Esta agrega el conjunto de descendientes en el sistema que están más cerca del argumento expandido y tienen el órden más alto.

Proposición 4.2.7. *Una expansión breadth first ranked es breadth first, debido al hecho de que todos los argumentos agregados son tan cercanos como cualquier otro al argumento expandido.*

Ejemplo 4.2.8. Ejemplificamos el comportamiento de una expansión breadth first ranked con el siguiente ejemplo en la figura 4.5, en el que la clasificación de argumentos está dada por alguna semántica $\leq$ para la cual $J > G \equiv H > I > C, D, E > F > A > B$.

Una expansión por caminos agrega un conjunto de atacantes no especificados en el marco que forman una ruta al argumento expandido.

Definición 4.2.9. Expansión por caminos. Una función de expansión de argumentos f_{DF} es llamada expansión por caminos si y solo si dado cualquier subgrafo $AF' = (A', R')$ y argumento $v \in A'$:

$$f_{DF}(AF, AF', v) = \{v_1, ..., v_n\} \text{ tal que } v_n = v, \\ \nexists v_0 \in A \text{ tal que } (v_0, v_1) \in R \text{ y } \forall\ 0 \leq i < n, (v_i, v_{i+1}) \in R$$

Dada una clasificación estricta (sin empates, eliminarlos es fácil y se puede hacer mediante órdenes predeterminados o basados en la identidad) $\leq$ sobre todos los argumentos de AF, definimos el siguiente esquema de expansiones.

Definición 4.2.10. Expansión por caminos ranked. Una función de expansión de argumentos $f_{DF,\leq}$ es llamada expansión por caminos ranked si y solo si dado cualquier subgrafo $AF' = (A', R')$ y argumento $v \in A'$:

$$f_{DF,\leq}(AF, AF', v) = \{(v_1, ..., v_n) \mid v_n = v, \nexists v_0 \in A \text{ tal que } (v_0, v_1) \in R, \\ \forall\ 0 \leq i < n, (v_i, v_{i+1}) \in R, \\ v_i \in A' \text{ implica } \forall (v_j, v_{i+1}) \in R, v_j \in A' \text{ y} \\ \forall v_j \text{ tal que } (v_j, v_{i+1}) \in R \text{ vale que } v_j \in A' \text{ o } v_i > v_j\}$$

Esta expansión agrega un camino en la cual cada argumento es el más alto clasificado con respecto a su sucesor.

Ejemplo 4.2.11. Ejemplifiquemos el comportamiento de las expansiones ranked por caminos con el siguiente ejemplo en la figura 4.6, en el que el ranking de argumentos está dado por alguna semántica $\leq$ para la cual $J > G > H > I > C > D > E > F > A > B$.

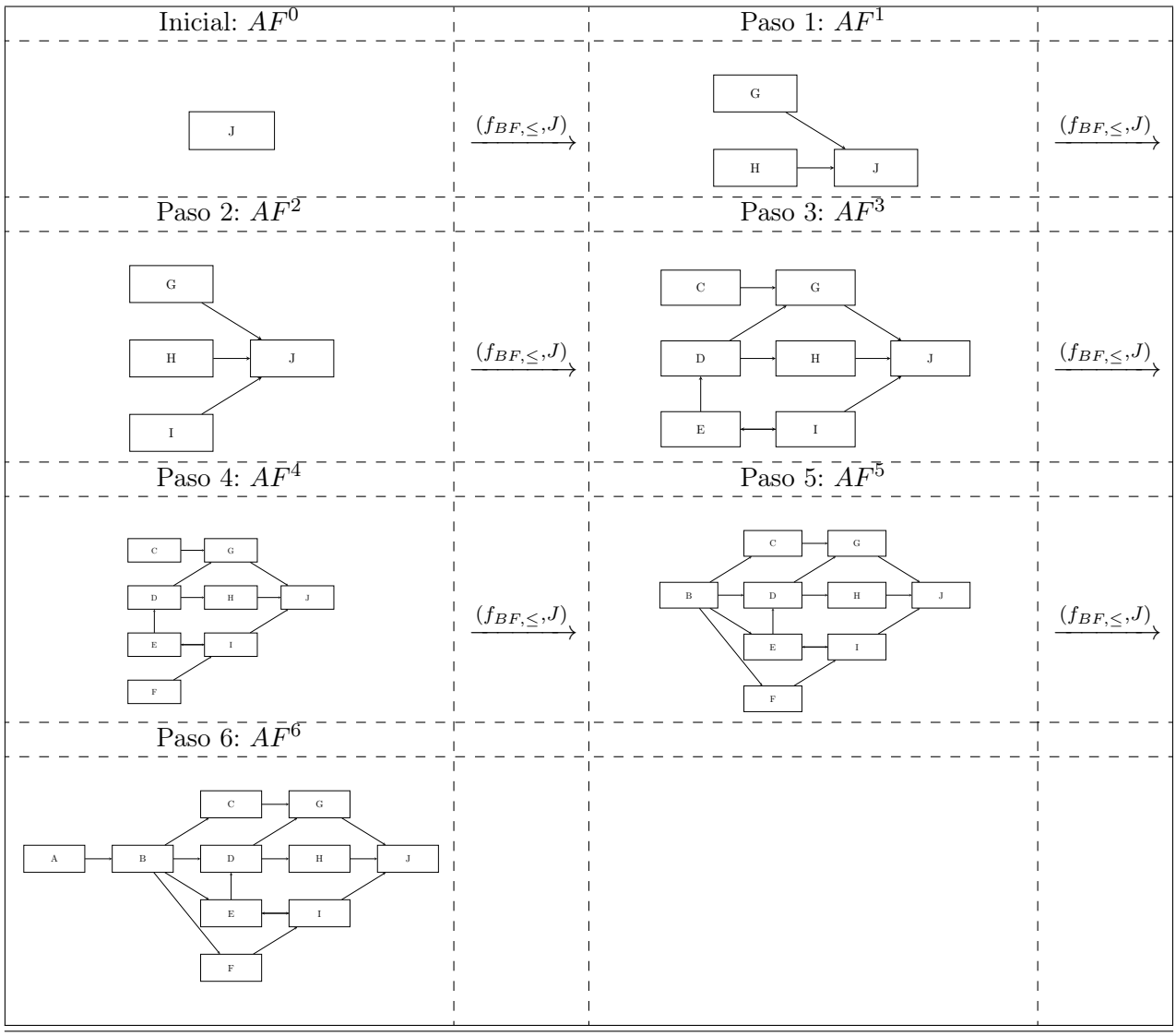

Fig. 4.5: Explicación breadth first ranked

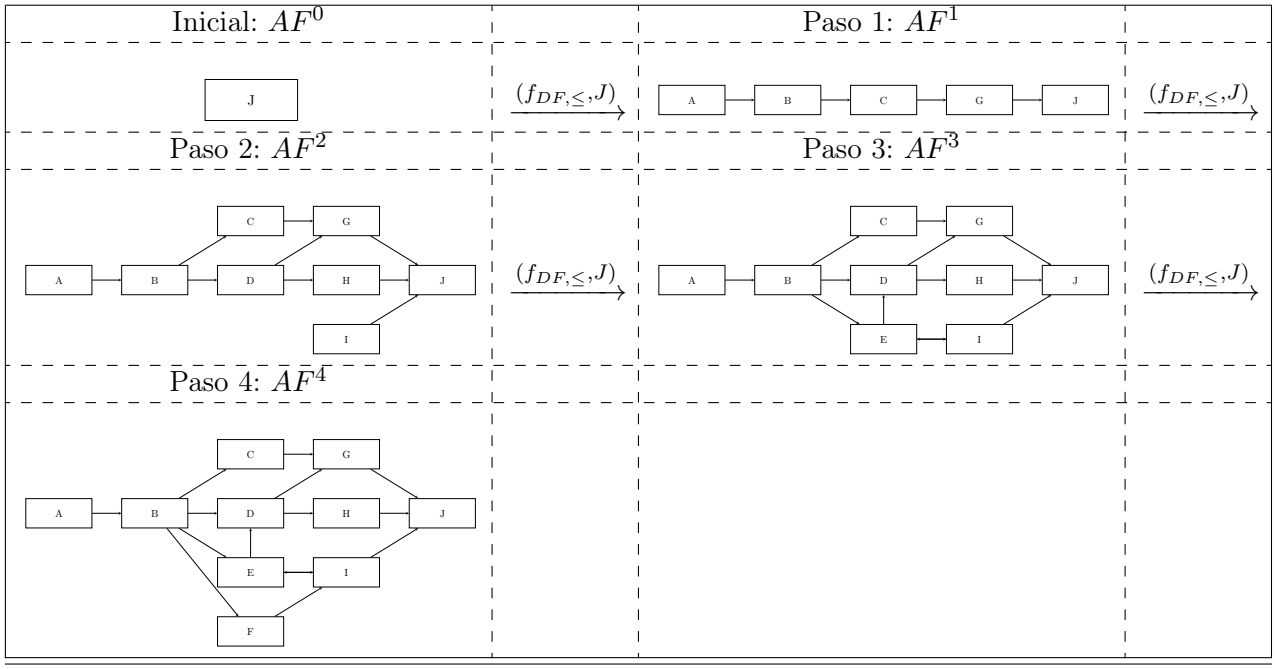

Fig. 4.6: Explicación por caminos ranked

4.3. Explicaciones en grafos de declaraciones

A continuación, redefinimos las nociones previas de explicaciones en el contexto de los grafos de declaraciones.

Definición 4.3.1. Expansión (de declaraciones). Dado un grafo declarativo $\mathcal{SG} = (\mathcal{V}, \mathcal{E}_S, \mathcal{E}_A)$, decimos que f es una función de expansión de sentencias si y solo si para cualquier subgrafo $\mathcal{SG}' = (\mathcal{V}', \mathcal{E}'_S, \mathcal{E}'_A)$ de $\mathcal{SG}$ y declaración $v \in V'$, $\mathcal{SG}' \xrightarrow{f(\mathcal{SG}, \mathcal{SG}', v)} \mathcal{SG}''$ es una expansión y se sostiene que:

- **(Inclusión)** $f(\mathcal{SG}, \mathcal{SG}', v) \subseteq \mathcal{V}$.
- **(Conectado)** $\forall v' \in f(\mathcal{SG}, \mathcal{SG}', v)$, existe un camino desde v' a v en $\mathcal{SG}''$ ya sea con ejes de soporte o ataque.
- **(Completo)** $\forall v' \in \mathcal{V}$ tal que existe un camino desde v' a v (ya sea con ejes de soporte o ataque) en $\mathcal{SG}$, hay un número finito de pasos $n \in \mathbb{N}$ tal que $\mathcal{SG}' = \mathcal{SG}^0 \xrightarrow{(f,v)} \mathcal{SG}^1 \xrightarrow{(f,v)} ... \xrightarrow{(f,v)} \mathcal{SG}^n = (\mathcal{V}^n, \mathcal{E}^n_S, \mathcal{E}^n_A)$ y $v' \in \mathcal{V}^n$.

Lo denotamos $\mathcal{SG}' \xrightarrow{(f,v)} \mathcal{SG}''$.

Definición 4.3.2. Explicación (de declaraciones). Dado un grafo declarativo $\mathcal{SG} = (\mathcal{V}, \mathcal{E}_S, \mathcal{E}_A)$, un etiquetado L, una función de expansión de declaraciones f y una declaración s, una explicación para s con respecto a $\mathcal{SG}$, L y f es una sucesión de expansiones:

$$\mathcal{SG}^0 \xrightarrow{(f,s_0)} \mathcal{SG}^1 \xrightarrow{(f,s_1)} ... \xrightarrow{(f,s_{n\text{-}1})} \mathcal{SG}^n$$

donde $\mathcal{SG}^0 = (\{s\}, \emptyset, \emptyset)$, cada s_i es una declaración de $\mathcal{SG}^i$ y $L_{\mathcal{SG}}(s) = L_{\mathcal{SG}^n}(s)$.

4.4. Esquemas de expansión en grafo declarativos

Además de los esquemas de expansión definidos previamente, definiremos los siguientes esquemas que tienen en cuenta la estructura de los grafos de declaración. En particular, harán uso de la noción de soportes y derivaciones para expandir una declaración dependiendo de sus premisas y las declaraciones atacando o apoyando las mismas. Además, utilizaremos las semánticas de clasificación definidas previamente para priorizar las partes más importantes del grafo con el fin de priorizar los motivos más fuertes detrás del estado de una determinada declaración.

Primero definamos la noción de distancia entre declaraciones en un grafo.

Definición 4.4.1. Distancia. Dado un grafo declarativo $\mathcal{SG} = (\mathcal{V}, \mathcal{E}_S, \mathcal{E}_A)$ y dos declaraciones $s, s' \in \mathcal{SG}$ tal que hay un camino de ejes de soporte y/o ataqueentre s y s' en $\mathcal{SG}$, la distancia entre s y s' es:

$$d_{\mathcal{SG}}(s, s') = \text{mín}\,\#(\{P \mid P \text{ es un camino de soportes y/o ataques desde } s \text{ a } s' \text{ en } \mathcal{SG}\})$$

Si no hay un camino de ataque entre s y s' en $\mathcal{SG}$ decimos que La distancia es infinita.

Dado un ranking $\leq$ sobre todas las declaraciones del grafo $\mathcal{SG} = (\mathcal{V}, \mathcal{E}_S, \mathcal{E}_A)$, definimos el siguiente esquema de expansión.

Definición 4.4.2. Expansión breadth first ranked (de declaraciones). Una función de expansión de declaraciones $f_{BF,\leq}$ es llamada breadth first ranked si y solo si dado cualquier subgrafo $\mathcal{SG}' = (\mathcal{V}', \mathcal{E}'_S, \mathcal{E}'_A)$ de $\mathcal{SG}$ y statement $s \in \mathcal{V}'$:

$$f_{BF,\leq}(\mathcal{SG}, \mathcal{SG}', s) = \{s' \mid \forall s'' \in (\mathcal{V} - \mathcal{V}') \text{ tal que } s', s'' \text{ ataca o soporta la misma premisa de } s \\ d_{\mathcal{SG}}(s, s') < d_{\mathcal{SG}}(s, s'') \text{ o } d_{\mathcal{SG}}(s, s') \equiv d_{\mathcal{SG}}(s, s'') \wedge s' \geq s''\}$$

Agrega el conjunto de descendientes en el grafo (por premisa) que están más cerca de la declaración expandida y tienen el rango más alto.

Ejemplo 4.4.3. Considerar el grafo declarativo de la figura 4.7. Luego, la figura 4.8 muestra una explicación breadth first ranked para $\leq$ como se ve para los pesos W mostrados.

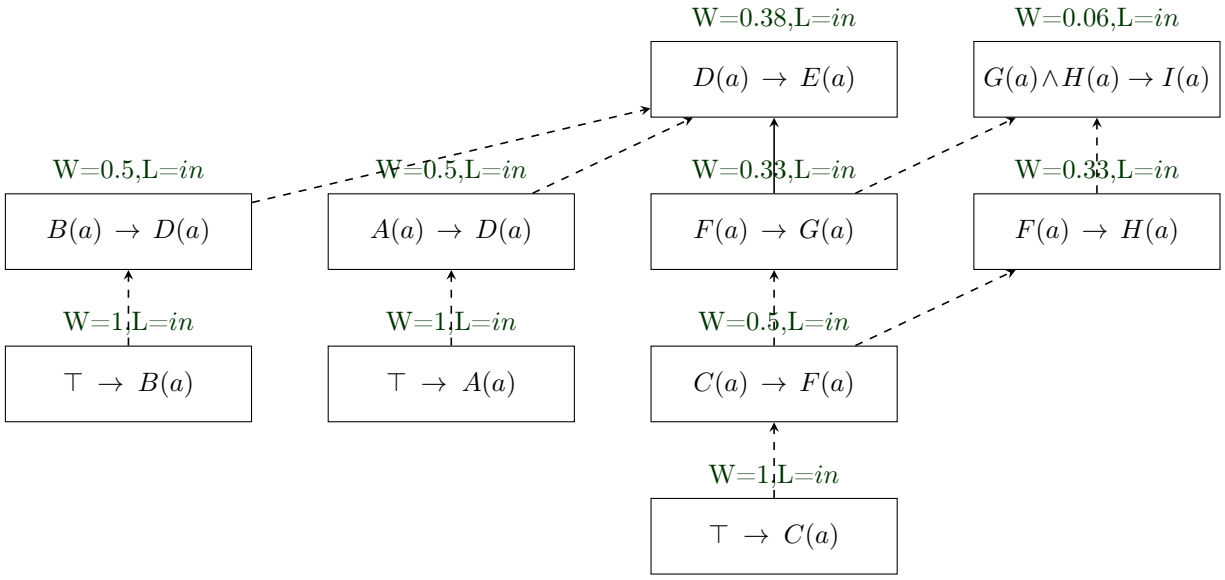

Fig. 4.7: Ejemplo grafo declarativo obteniendo los pesos mediante acumulador *Cat* intra-premisa

Dado un ranking estricto (sin empates, quitarlos es fácil, por ejemplo mediante su identidad) $\leq$ sobre todas las declaraciones de $\mathcal{SG}$, definimos el siguiente esquema de expansión.

Definición 4.4.4. Expansión path ranked (de declaraciones). Una función de expansión de declaraciones $f_{DF,\leq}$ es llamada path ranked si y solo si dado cualquier subgrafo $\mathcal{SG}' = (\mathcal{V}', \mathcal{E}'_S, \mathcal{E}'_A)$ de $\mathcal{SG}$ y declaración $s \in \mathcal{V}'$:

$$f_{PR,\leq}(\mathcal{SG}, \mathcal{SG}', s) = K, \text{donde } K \text{ es el conjunto mínimo tal que}$$

- $s \in K$,
- $\forall s' \in K, \forall p_i \in Premise(s')$ vale que
  - $\forall s'' \in \mathcal{V}$ tal que s'' ataca o soporta p_i, $s'' \in \mathcal{V}'$, o
  - $\exists! s'' \in K$ tal que s'' ataca o soporta p_i y $\forall s''' \in \mathcal{V}$ que ataca o soporta p_i: $s'' > s'''$

Esta expansión agrega la declaración más fuerte que admite o ataca cada premisa de la declaración original y también sigue asi recursivamente hasta que alcanza declaraciones con todos sus ataques y soportes dentro del grafo actual.

Ejemplo 4.4.5. La siguiente figura 4.9 muestra una explicación path ranked donde $\leq$ es dado por los pesos W que se muestran.

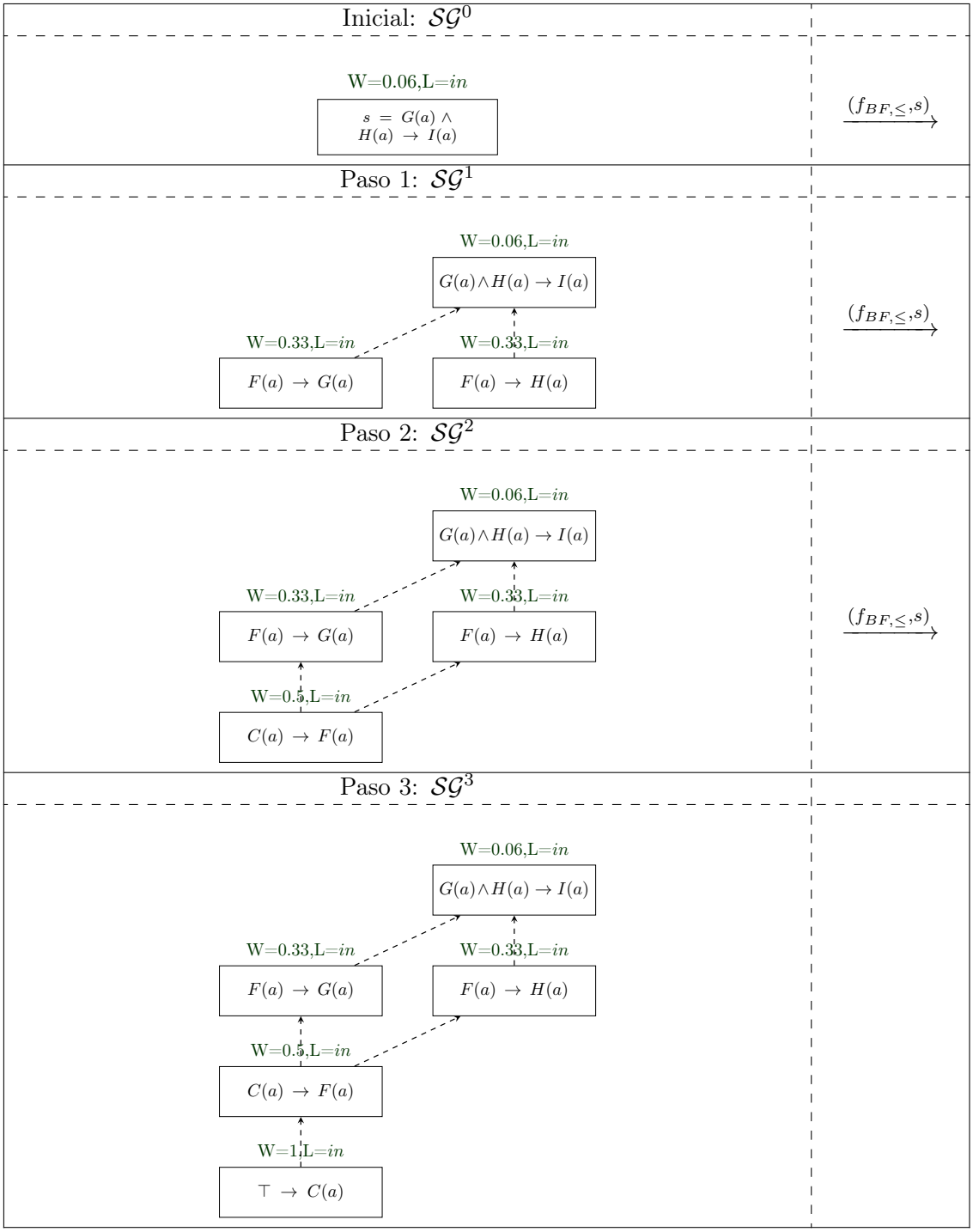

Fig. 4.8: Explicación breadth first ranked

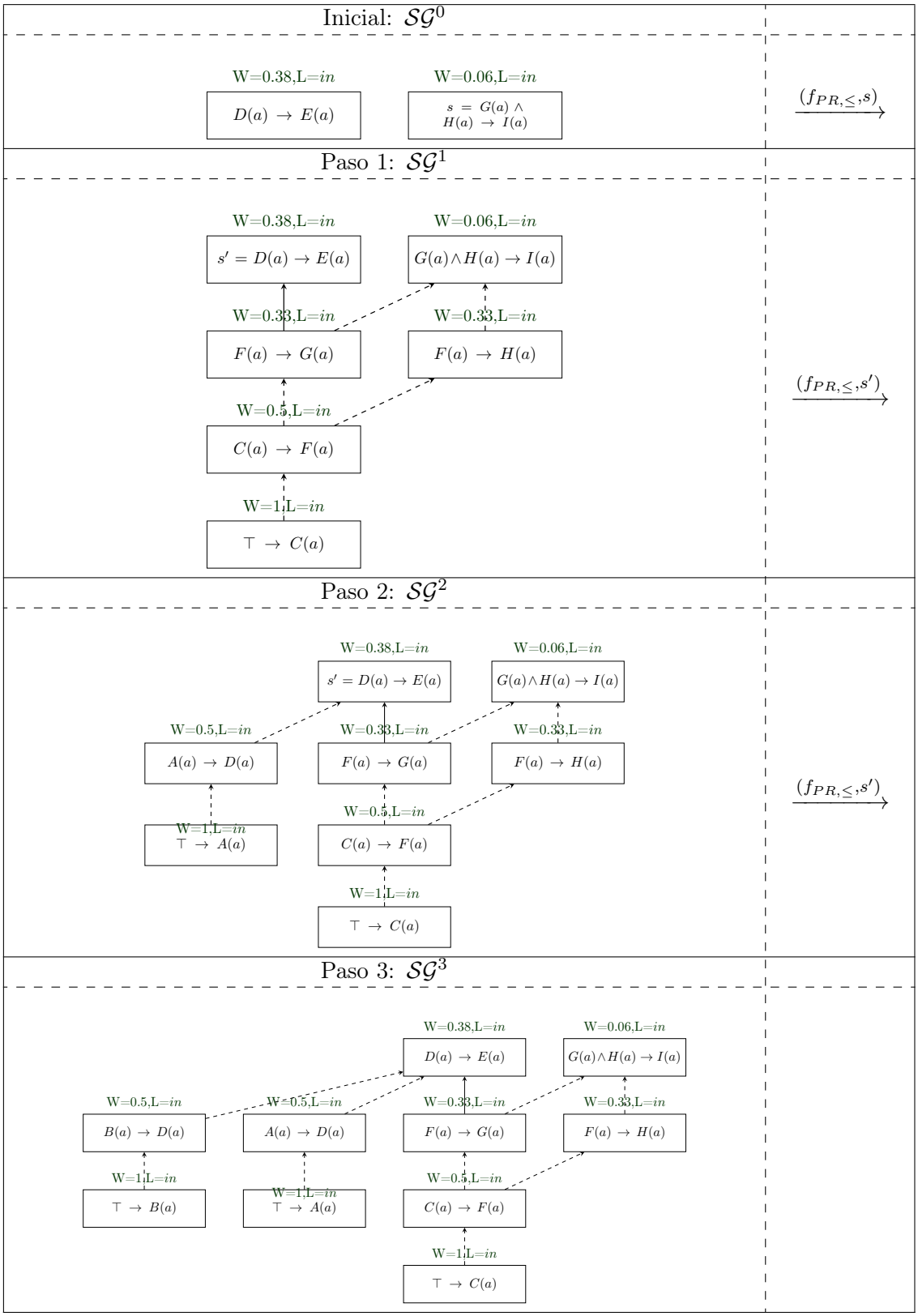

Fig. 4.9: Explicación path ranked

5. CONCLUSIONES Y TRABAJO FUTURO

En este trabajo, presentamos dos marcos de razonamiento principales: argumentación y lógicas rebatibles a través de grafos de declaraciones.

Ambos sistemas definen formas de razonamiento ante la contradicción, en el caso de la argumentación, a través de extensiones y semántica de clasificación, mientras que los grafos de declaraciones utilizan funciones de etiquetado para representar los diferentes sabores / intuiciones de las lógicas desarmables según lo discutido por los lógicos.

Luego, desarrollamos semánticas de ranking en el contexto de grafos declarativos como en la argumentación abstracta para cuantificar la fuerza de las declaraciones (que contienen reglas lógicas). Nuestra semántica de clasificación para grafos de declaraciones tiene en cuenta el contenido de los nodos en el grafo para proporcionar cuantificaciones que reflejen la estructura de las fórmulas.

La semántica de clasificación que presentamos debe estudiarse más a fondo para caracterizarlos más adecuadamente y desarrollar diferentes principios para poder compararnos. Al mismo tiempo, nos gustaría diseñar semánticas de ranking basadas en probabilidad teniendo en cuenta nociones de lógicas probabilísticas y subjetivas.

Otro tema de trabajo adicional debería ser estudiar la convergencia de las semánticas de ranking presentadas para los grafos declarativos en presencia de ciclos. Existe un trabajo existente (pero no sustancial) en la convergencia de las semánticas de ranking para sistemas de argumentación abstractos sobre los cuales podríamos elaborar.

Además, la principal contribución presentada es una metodología para explicar y aumentar progresivamente el conocimiento a un receptor humano de tal manera que no lo abrume la información innecesaria. Por esta razón, las semánticas de ranking fueron utilizadas como técnica para priorizar los factores clave que conducen a conclusiones, en presencia de múltiples caminos de razonamiento que conducen a lo mismo.

Aún así, hay muchos puntos a seguir como investigación futura. Es decir, queremos proporcionar medidas de efectividad y realizar un experimento a gran escala para confirmar nuestras intuiciones. Además, nos gustaría probar en qué contextos (enseñanza, persuasión y aclaración) nuestra explicación sería la más útil.

Finalmente, queremos definir más mecanismos de expansión y probarlos frente a grandes bases de conocimiento, particularmente frente a problemas de explosión combinatoria.

Bibliografía

[1] Leila Amgoud and Jonathan Ben-Naim. Ranking-based semantics for argumentation frameworks. In *International Conference on Scalable Uncertainty Management*, pages 134–147. Springer, 2013.

[2] Leila Amgoud and Jonathan Ben-Naim. Weighted bipolar argumentation graphs: Axioms and semantics. In *IJCAI*, pages 5194–5198, 2018.

[3] Grigoris Antoniou. Defeasible reasoning: A discussion of some intuitions. *International journal of intelligent systems*, 21(6):545–558, 2006.

[4] Abdallah Arioua and Madalina Croitoru. Formalizing explanatory dialogues. In *International Conference on Scalable Uncertainty Management*, pages 282–297. Springer, 2015.

[5] Jean-François Baget, Fabien Garreau, Marie-Laure Mugnier, and Swan Rocher. Revisiting chase termination for existential rules and their extension to nonmonotonic negation. *arXiv preprint arXiv:1405.1071*, 2014.

[6] Jean-François Baget, Michel Leclère, Marie-Laure Mugnier, and Eric Salvat. On rules with existential variables: Walking the decidability line. *Artificial Intelligence*, 175(9-10):1620–1654, 2011.

[7] Jean-François Baget, Marie-Laure Mugnier, Sebastian Rudolph, and Michaël Thomazo. Walking the complexity lines for generalized guarded existential rules. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[8] Catriel Beeri and Moshe Y Vardi. The implication problem for data dependencies. In *International Colloquium on Automata, Languages, and Programming*, pages 73–85. Springer, 1981.

[9] Philippe Besnard and Anthony Hunter. A logic-based theory of deductive arguments. *Artificial Intelligence*, 128(1-2):203–235, 2001.

[10] David Billington. Defeasible logic is stable. 1993.

[11] Elise Bonzon, Jérôme Delobelle, Sébastien Konieczny, and Nicolas Maudet. A comparative study of ranking-based semantics for abstract argumentation. *CoRR*, abs/1602.01059, 2016.

[12] Gerhard Brewka and Stefan Woltran. Abstract dialectical frameworks. In *Twelfth International Conference on the Principles of Knowledge Representation and Reasoning*, 2010.

[13] Martin Caminada. On the issue of reinstatement in argumentation. In *European Workshop on Logics in Artificial Intelligence*, pages 111–123. Springer, 2006.

[14] Claudette Cayrol and Marie-Christine Lagasquie-Schiex. Gradual acceptability in argumentation systems. In *Proc. of the 3rd CMNA (International workshop on computational models of natural argument)*, pages 55–58. Citeseer, 2003.

[15] Phan Minh Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial intelligence*, 77(2):321–357, 1995.

[16] Marcelo A Falappa, Gabriele Kern-Isberner, and Guillermo R Simari. Explanations, belief revision and defeasible reasoning. *Artificial Intelligence*, 141(1-2):1–28, 2002.

[17] Alejandro J García, Carlos I Chesñevar, Nicolás D Rotstein, and Guillermo R Simari. Formalizing dialectical explanation support for argument-based reasoning in knowledge-based systems. *Expert Systems with Applications*, 40(8):3233–3247, 2013.

[18] Abdelraouf Hecham. *Defeasible reasoning for existential rules*. Theses, Université de Montpellier, July 2018.

[19] Abdelraouf Hecham, Pierre Bisquert, and Madalina Croitoru. On a Flexible Representation for Defeasible Reasoning Variants. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1123–1131. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[20] John F Horty, DS Touretzky, and RH Thomason. A clash of intuitions: the current state of nonmonotonic multiple inheritance systems. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 476–482. Morgan Kaufmann Los Altos, CA, 1987.

[21] Frederick Maier and Donald Nute. Well-founded semantics for defeasible logic. *Synthese*, 176(2):243–274, 2010.

[22] David Makinson and Karl Schlechta. Floating conclusions and zombie paths: two deep difficulties in the "directly skeptical" approach to defeasible inheritance nets. *Artificial intelligence*, 48(2):199–209, 1991.

[23] Bruno Marnette. Generalized schema-mappings: from termination to tractability. In *Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 13–22. ACM, 2009.

[24] Paul-Amaury Matt and Francesca Toni. A game-theoretic measure of argument strength for abstract argumentation. In *European Workshop on Logics in Artificial Intelligence*, pages 285–297. Springer, 2008.

[25] Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 2018.

[26] Tim Miller, Piers Howe, and Liz Sonenberg. Explainable ai: Beware of inmates running the asylum or: How i learnt to stop worrying and love the social and behavioural sciences. *arXiv preprint arXiv:1712.00547*, 2017.

[27] Bernard Moulin, Hengameh Irandoust, Micheline Bélanger, and Gaëlle Desbordes. Explanation and argumentation capabilities: Towards the creation of more persuasive agents. *Artificial Intelligence Review*, 17(3):169–222, 2002.

[28] John L Pollock. Defeasible reasoning. *Cognitive science*, 11(4):481–518, 1987.

[29] Henry Prakken. Intuitions and the modelling of defeasible reasoning: some case studies. *arXiv preprint cs/0207031*, 2002.

[30] Fuan Pu, Jian Luo, Yulai Zhang, and Guiming Luo. Argument ranking with categoriser function. In *International Conference on Knowledge Science, Engineering and Management*, pages 290–301. Springer, 2014.

[31] Richard W Southwick. Explaining reasoning: an overview of explanation in knowledge-based systems. *The knowledge engineering review*, 6(1):1–19, 1991.