1.2 Reasoning Framework for the Winograd Schema Challenge

We developed a logical reasoning based framework for solving the WSC problems. The framework takes the formal representations of a WSC problem (a sentence and a question) and a knwoledge instance as inputs and performs a series of logical operations and intermediate generations to deduce the answer to the question. The inputs and the set of intermediate operations are as described below.

There are three inputs to the reasoning framework, namely, a winograd schema sentence, a corresponding winograd schema question and a commonsense knowledge instance which belongs to any one of the first ten knowledge types defined in the previous section. In this section, we show how, through a series of operations, the three inputs to the reasoning component entail the answer(s) to the input question. To show that, we define the inputs formally and also define a set of operations and the intermediate results.

1.2.1 Defining Inputs to the Reasoning Framework

Definition 1 (Formal Representation of the Given Text) *A formal representation of a given text ($\mathcal{T}$) is a rooted, edge labeled directed acyclic graph, $\mathcal{G}(\mathcal{N},\mathcal{E})$, consisting of the set $\mathcal{N}$ of nodes and the set $\mathcal{E}$ of edges, where $\mathcal{E}$ contains the edge labels between the ordered pairs of elements of $\mathcal{N}$.*

The set of nodes in $\mathcal{G}$ consists of two types of nodes, i.e., $\mathcal{N} = \mathcal{C} \bigcup \mathcal{C}_o$, where

- *$\mathcal{C}$ is a set of constants in $\mathcal{T}$. The constants are the actual words [1] (uni-grams) in $\mathcal{T}$ that are nouns, verbs, adjectives, adverbs and pronouns, and the Wh question words in $\mathcal{T}$. In other words, $\mathcal{C} = \mathcal{C}_V \bigcup \mathcal{C}_N \bigcup \mathcal{C}_P \bigcup \mathcal{C}_R \bigcup \mathcal{C}_J \bigcup \mathcal{C}_W$, where*
  - *$\mathcal{C}_V$ is a set of verbs in $\mathcal{T}$,*
  - *$\mathcal{C}_N$ is a set of nouns in $\mathcal{T}$,*

[1] The idea of constant nodes is borrowed from (Bos, 2016) about AMR representation (Banarescu *et al.*, 2013)

- $\mathcal{C}_P$ *is a set of pronouns in* $\mathcal{T}$*,*
- $\mathcal{C}_R$ *is a set of adverbs in* $\mathcal{T}$*,*
- $\mathcal{C}_J$ *is a set of adjectives in* $\mathcal{T}$*, and*
- $\mathcal{C}_W = \{q_1, q_2, ..q_n\}$ *such that "*q_i*" is a placeholder for an unknown entity* [2] *in* $\mathcal{T}$*.*

Constants' occurrence index in $\mathcal{T}$ *is also preserved (except for "*q_i*"s).*

- $\mathcal{C}_o$ *is a set of concepts to which the constants in* $\mathcal{C}$ *belong. The set of concepts is divided into two categories, namely* $\mathcal{C}_{o-N}$ *and* $non - \mathcal{C}_{o-N}$*.*
  - *The set* $\mathcal{C}_{o-N}$ *consists of the conceptual classes of the nouns and pronouns in* $\mathcal{C}$*. Each item in* $\mathcal{C}_{o-N}$ *represents a basic class of entities to which a particular noun or a pronoun belongs. There may be more than one conceptual class for an element in* $\mathcal{C}$*.* $\mathcal{C}_{o-N}$ *also contains "*q*" which represents the conceptual class of an unknown entity identifier (i.e., "*q_i*"). An unknown entity identifier always belongs to at-least two conceptual classes, one is the real concept that it belongs to, and the other is "*q*".*
  - *The set* $non - \mathcal{C}_{o-N}$ *consists of the base form of the words (except nouns and pronouns) in* $\mathcal{C}$*.*

The set of edge labels in $\mathcal{G}$ *consists of two kinds of edges i.e.,* $\mathcal{E} = \mathcal{E}_{CC_o} \bigcup \mathcal{E}_{CC}$

- $\mathcal{E}_{CC_o}$ *is a set of edge labels that are used to represent the edges from a node in* $\mathcal{C}$ *to a node in* $\mathcal{C}_o$*. In other words, these edge labels are used to represent the relationship between a constant node in* $\mathcal{G}$ *to a concept node in* $\mathcal{G}$*. There is only one such relationship possible in* $\mathcal{G}$*, i.e.* $\mathcal{E}_{CC_o} = \{instance\_of\}$*.*

[2] An unknown entity is represented by any one of the following words in an English question. $what, when, where, which, who, whom$, and $whose$

- $\mathcal{E}_{CC}$ *is a set of edge labels that are used to represent the edges that originate from and end in the nodes in* $\mathcal{C}$*. In other words, these edge labels are used to represent the relationship between two constant nodes in* $\mathcal{G}$*. Based on the type of the end nodes, the edge labels are categorized into seven kinds i.e.,*

  $\mathcal{E}_{CC} = \mathcal{E}_{CC(V-V)} \bigcup \mathcal{E}_{CC(V-N/P/W)} \bigcup \mathcal{E}_{CC(V-J)} \bigcup \mathcal{E}_{CC(J-V)} \bigcup \mathcal{E}_{CC(V-R)} \bigcup \mathcal{E}_{CC(N-N/P/W)} \bigcup \mathcal{E}_{CC(N/P/W-J)}$

  - $\mathcal{E}_{CC(V-V)}$ *represents the set of all the possible edge labels from and to items in* $\mathcal{C}_V$*.*
  - $\mathcal{E}_{CC(V-N/P/W)}$ *represents the set of all the possible edge labels from items in* $\mathcal{C}_V$ *to the items in* $\mathcal{C}_N$ *or* $\mathcal{C}_P$ *or* $\mathcal{C}_W$*.*
  - $\mathcal{E}_{CC(V-J)}$ *represents the set of all the possible edge labels from items in* $\mathcal{C}_V$ *to the items in* $\mathcal{C}_J$*.*
  - $\mathcal{E}_{CC(J-V)}$ *represents the set of all the possible edge labels from items in* $\mathcal{C}_J$ *to the items in* $\mathcal{C}_V$*.*
  - $\mathcal{E}_{CC(V-R)}$ *represents the set of all the possible edge labels from items in* $\mathcal{C}_V$ *to the items in* $\mathcal{C}_R$*.*
  - $\mathcal{E}_{CC(N-N/P/W)}$ *represents the set of all the possible edge labels from items in* $\mathcal{C}_N$ *to the items in* $\mathcal{C}_N$ *or* $\mathcal{C}_P$ *or* $\mathcal{C}_W$*.*
  - $\mathcal{E}_{CC(N/P/W-J)}$ *represents the set of all the possible edge labels from items in* $\mathcal{C}_N$ *or* $\mathcal{C}_P$ *or* $\mathcal{C}_W$ *to the items in* $\mathcal{C}_J$*.*

  The items in $\mathcal{E}_{CC}$*, consequently its various subcategories, consist of a predefined set of semantic relations. The semantic relations for representing* $\mathcal{G}$ *could come from the vocabulary of any semantic parser, provided they satisfy the definition of* $\mathcal{G}$*. For this work we refer to the semantic relations in an off the shelf semantic* *parser callef Knowledge Parser (K-Parser)* [3] . *For each of the edge label (e_i) in $\mathcal{E}_{CC} - \{\mathcal{E}_{CC(V-J)}, \mathcal{E}_{CC(J-V)}\}$, there also exists an inverse edge label e_i^{-1} such that the direction of the edges represented by e_i and e_i^{-1} is opposite in $\mathcal{G}$. For example if $e_i = instance\_of$ then $e_i^{-1} = instance$.*

$\mathcal{E}_{CC}^{-1} = \mathcal{E}_{CC(V-V)}^{-1} \bigcup \mathcal{E}_{CC(V-N/P)}^{-1} \bigcup \mathcal{E}_{CC(V-R)}^{-1}, \mathcal{E}_{CC(N-N/P)}^{-1} \bigcup \mathcal{E}_{CC(N/P/W-J)}^{-1}$ *represents the set of inverse edge labels in $\mathcal{G}$.*

The examples of each attribute of the definition above are as mentioned below.

- the word *"man"* occurs at index two in the example shown in Figure 1.1 and it is represented as *"man_2"* in the formal representation of the sentence.
- a *"man"* is a *"person"* is shown in Figure 1.1 with the help of the nodes *"man_2"* and *"person"*
- "q_1" in Figure 1.2 is a *"person"* is shown with the help of the nodes *"q_1"* and *"person"*
- the edge labeled *"instance_of"* from the node *"man_2"* to the node *"person"* in Figure 1.1 represents a constant to concept edge

Definition 2 (Representation of the Input Sentence ($\mathcal{G}_S$)) *A representation of the input English sentence S is a graph of the form $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$ (from definition 1), with following constraints.*

- *$\mathcal{N}$ is the set of nodes extracted from S, such that there are no unknown nodes in $\mathcal{N}$, i.e., $\mathcal{C}_W = \emptyset$*
- *$\mathcal{E}$ is the set of edges between the nodes in $\mathcal{N}$.*

[3] made available by its developers at www.kparser.org

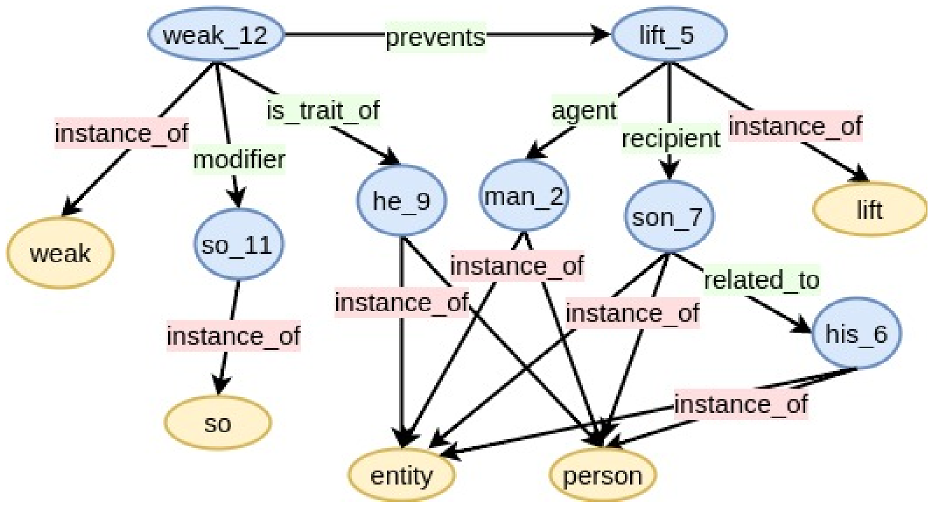

Figure 1.1: Graphical Representation of an input sentence, *"The man could not lift his son because he was so weak."*

An example of the representation of an input sentence is shown in Figure 1.1.

Definition 3 (Representation of the Input Question ($\mathcal{G}_Q$)) *A representation of the input English question Q is a graph of the form $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$ (from definition 1), with following constraints.*

- *$\mathcal{N}$ is the set of nodes extracted from Q, such that $\mathcal{N}$ contains at least one unknown node, i.e., $\mathcal{C}_W \neq \emptyset$*
- *$\mathcal{E}$ is the set of edges between the nodes in $\mathcal{N}$.*

An example of the represent of an input question is shown in Figure 1.2

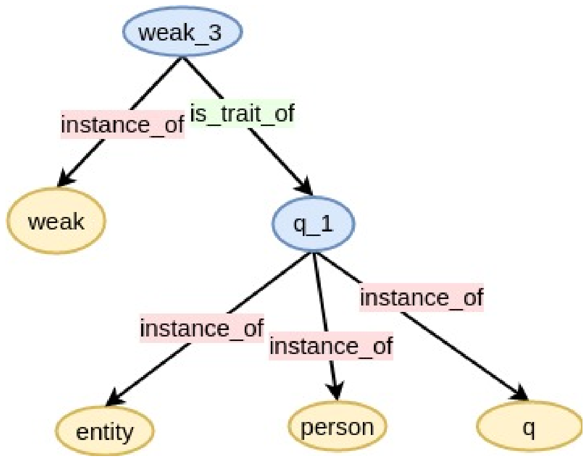

Figure 1.2: Graphical Representation of an Input Question, *“Who was weak?”*

Representation of an Input Knowledge Instance

The representation of the commonsense knowledge is a graphical transformation of the categories of the commonsense knowledge defined in the previous section. Following definitions present the different aspects of the transformation.

Definition 4 (Transformation of an Entity) *The generalized representation of an entity, as defined in the previous section, is shown below.*

$$\mathtt{x}\ \mathtt{[instance\_of:}\ \mathtt{C,}$$
$$\mathtt{trait:}\ \mathtt{p}_1\mathtt{,\ ...,}\ \mathtt{trait:}\ \mathtt{p}_n\mathtt{,}$$
$$\mathtt{rel}_1\mathtt{:}\ \mathtt{x}_1\mathtt{,\ ...,}\ \mathtt{rel}_m\mathtt{:}\ \mathtt{x}_m\mathtt{]}$$

where, $\mathtt{x}$ *is a placeholder that represents an entity,* $\mathtt{C}$ *represents the type of the entity,* $\mathtt{p}_1\mathtt{,\ ...,}\ \mathtt{p}_n$ *represent the properties associated with x via a relation* $\mathtt{trait}$ *and* $\mathtt{x}_1\mathtt{,\ ...,}$ $\mathtt{x}_n$ *represent the entities that are associated with x via the relations* $\mathtt{rel}_1\mathtt{,\ ...,}\ \mathtt{rel}_m$ *respectively.*

The above representation is transformed into a rooted, edge labeled directed acyclic graph $\mathcal{G}_{Ent} = \{\mathcal{N}_{Ent}, \mathcal{E}_{Ent}\}$ *such that* $\mathcal{N}_{Ent}$ *represents the set of nodes in* $\mathcal{G}_{Ent}$ *and* $\mathcal{E}_{Ent}$

represents the set of edge labels in $\mathcal{G}_{Ent}$. Following steps are taken to transform the above representation into $\mathcal{G}_{Ent}$.

(i) x *is transformed into the root node with label* x*,*

(ii) C *is transformed into a non-root node with label* C*.*

(iii) The root nodes of the entities $x_1, \ldots, x_n$*, defined recursively, are connected to the root node via directed relations* $rel_1, \ldots, rel_m$ *from the root* x *to each entity's root node respectively.*

(iv) Same as the representation of an entity the properties are also represented as rooted, edge labeled directed acyclic graphs (see the section Transformation of a Property below). The root nodes of the properties $p_1, \ldots, p_m$ *are connected to the root node via directed relation* $trait$ *from the root* x *to each property's root node respectively.*

The graph generated for the above generalized representation is shown in Figure 1.3.

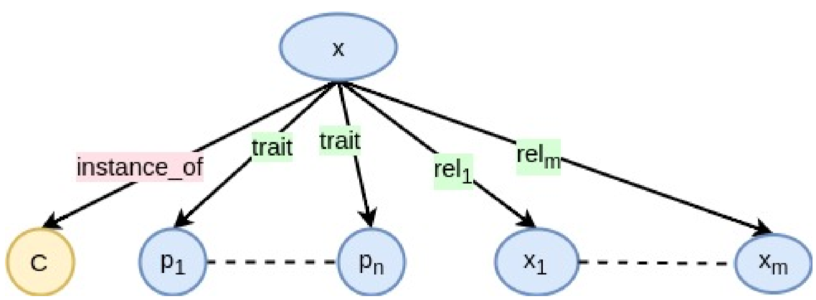

Figure 1.3: General Graphical Representation of an Entity

Definition 5 (Transformation of a Property) *The generalized representation of a property, as defined in the previous section, is shown below.*

$$p\ [instance\_of:\ PROP,\ is\_trait\_of:\ x_1, \ldots,\ is\_trait\_of:\ x_n]$$

where, p *is a placeholder that represents a property,* $PROP$ *represents the actual value of the property and* $x_1, \ldots, x_n$ *represent the entities that are associated with p via* is_trait_of *relation.*

The above representation is transformed into a rooted, edge labeled directed acyclic graph $\mathcal{G}_{Prop} = \{\mathcal{N}_{Prop}, \mathcal{E}_{Prop}\}$ *such that* $\mathcal{N}_{Prop}$ *represents the set of nodes in* $\mathcal{G}_{Prop}$ *and* $\mathcal{E}_{Prop}$ *represents the set of edge labels in* $\mathcal{G}_{Prop}$*. Following steps are taken to transform the above representation into* $\mathcal{G}_{Prop}$*.*

(i) p *is transformed into the root node with label* p*,*

(ii) $PROP$ *is transformed into a non-root node with label* $PROP$*.*

(iii) The root nodes of the entities $x_1, \ldots, x_n$*, as defined in the section Transformation of an Entity above, are connected to the root node* p *via directed relation* is_trait_of *from the root node* p *to each property's root node respectively.*

The graph generated from the above generalized representation is shown in Figure 1.4.

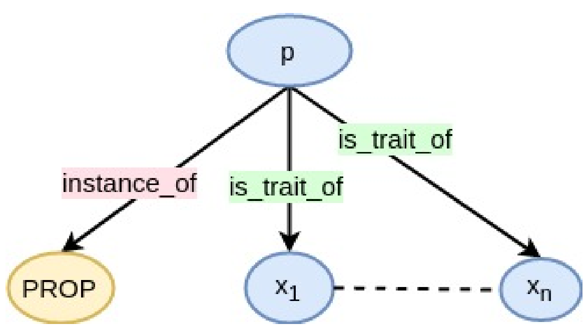

Figure 1.4: General Graphical Representation of a Property

Definition 6 (Transformation of an Event) *The generalized representation of an event, as defined in the previous section, is shown below.*

$$e\ [instance\_of:\ A,\ rel_1:\ x_1,\ldots,\ rel_n:\ x_n]$$

where, e is a placeholder that represents an event, A represents the driver action of the event and $x_1, \ldots, x_n$ represent the entities that are participating in the event via relations/roles $rel_1, \ldots, rel_n$ respectively.

The above representation is transformed into a rooted, edge labeled directed acyclic graph $\mathcal{G}_E = \{\mathcal{N}_E, \mathcal{E}_E\}$ such that $\mathcal{N}_E$ represents the set of nodes in $\mathcal{G}_E$ and $\mathcal{E}_E$ represents the set of edge labels in $\mathcal{G}_E$. Following steps are taken to transform the above representation into $\mathcal{G}_E$.

(i) *e is transformed into the root node with label e,*

(ii) *A is transformed into a non-root node with label A.*

(iii) *The root nodes of the entities $x_1, \ldots, x_n$, as defined above in the section Transformation of an Entity, are connected to the root node e via directed relations $rel_1, \ldots, rel_n$ from the root e to each entity's root node respectively.*

The graph generated from the above generalized representation is shown in Figure 1.5.

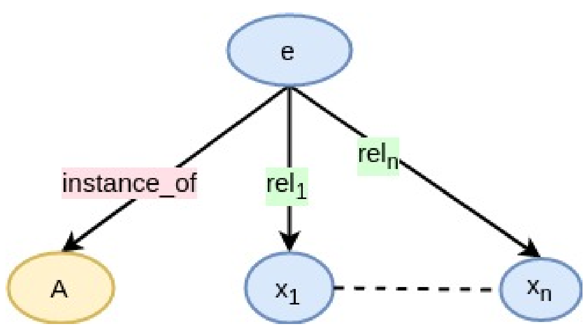

Figure 1.5: General Graphical Representation of An Event

Definition 7 (Transformation of the Execution of an Event) *As mentioned in the previous section, there are two possible scenarios for the execution of an event. An event may be executed or not. In the generalized representation, there aspects are represented by the symbols* `[+]` *and* `[-]` *respectively. Whereas nothing is added while transforming the* `[+]` *into the graphical representation, the* `[-]` *symbol is transformed into an outgoing edge labeled negative from the driver action of the event to a newly created entity node of type negation.*

For example, let e be an event such that in a knowledge category e is represented as follow.

$$\texttt{[-] e [instance\_of: A, rel}_1\texttt{: x}_1\texttt{,..., rel}_n\texttt{: x}_n\texttt{]}$$

where, A is the driver action of e and $x_1, \ldots, x_n$ are entities participating in e, in the roles $rel_1, \ldots, rel_n$ respectively.

The above representation is transformed into the graphical representation shown in Figure 1.6

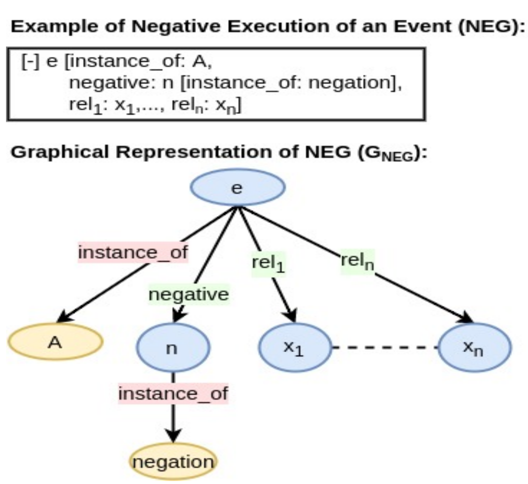

Figure 1.6: Graphical Representation of a Not Executed Event in Knowledge

Definition 8 (Transformation of Causal, Sequential and Co-occurrence Relations) *The causal, sequential and co-occurrence relations (causes, prevents, followed by, and and) exist between the events and properties in the different commonsense categories defined in this work (see previous section). These relations are directly transformed into directed edges between the roots of event graphs and property graphs as defined above.*

For example, let us consider the initial part (IP) (the part before the `implies` keyword) of an example knowledge instance from the commonsense category 1, as shown below.

```
p [instance-of: weak,
   is_trait_of: x [instance-of:person,
                   instance-of:entity]]
prevents
e [instance_of: lift;
   agent: y [instance-of:person,
             instance-of:entity]]
```

According to the transformation process of the causal and sequential relations, the above representation is transformed into a rooted, edge labeled directed acyclic graph $\mathcal{G}_{K-Cond}$ as shown in Figure 1.7.

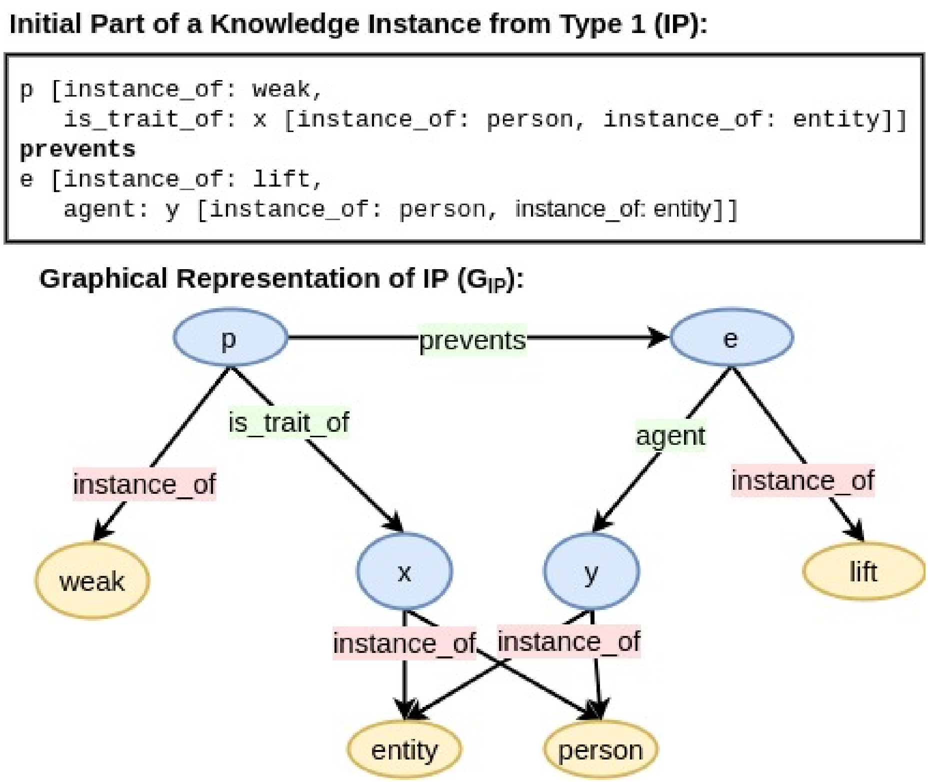

Figure 1.7: Graphical representation of the initial part of a knowledge instance from category 1

Definition 9 (Transformation of the Commonsense Knowledge Format) *As mentioned in the previous section, all the commonsense knowledge categories defined in this work are of the form below.*

```
S implies x=y
```

where S is a combination of a set of events and/or properties connected via a chain of sequential and causal relations. Both x *and* y *are entities that either participate in two events in* S *or in an event and a property in* S*.*

The above representation is transformed into a rooted, edge labeled directed acyclic *graph $\mathcal{G}_K = \{\mathcal{N}_K, \mathcal{E}_K\}$ such that $\mathcal{N}_K$ represents the set of nodes in $\mathcal{G}_K$, and $\mathcal{E}_K$ represents the set of edge labels in $\mathcal{G}_K$. Following steps are taken to transform the above representation into $\mathcal{G}_K$.*

(i) Each property in $\mathtt{S}$ is transformed into a graphical representation by following the steps in the Definition 5 to transform a property into a graph.

(ii) Each event in $\mathtt{S}$ is transformed into a graphical representation by following the steps in the Definition 6 to transform an event into a graph.

(iii) The sequential and causal relationships in $\mathtt{S}$ are established between the graphical representations of events and properties in $\mathtt{S}$ by following the steps in the Definition 8 for the transformation of causal and sequential relations.

(iv) Finally, the right hand side of the implication mentioned above is encoded in the graphical representation retrieved from the above steps by replacing either the entity $\mathtt{x}$ by $\mathtt{y}$ or vice-versa such that only one of them exists in the final graphical representation of a knowledge instance.

For example if the graph shown in Figure 1.7 represents S and the right side of the implication above is x=y then the root nodes of both the entities x and y are merged into one (keeping any one of the two labels) such that now all the incoming edges to both the nodes are going to just one node and all the outgoing node are coming from just one node i.e., the merged node. Also, the redundant edges and nodes which were created due to children of the roots of the entities x and y, are removed. An example transformation is as shown in Figure 1.8.

1.2.2 Defining Intermediate Operations of the Reasoning Framework

The following definitions establish some of the properties on the basis of the representations of the input sentence, the input question and the input commonsense knowledge.

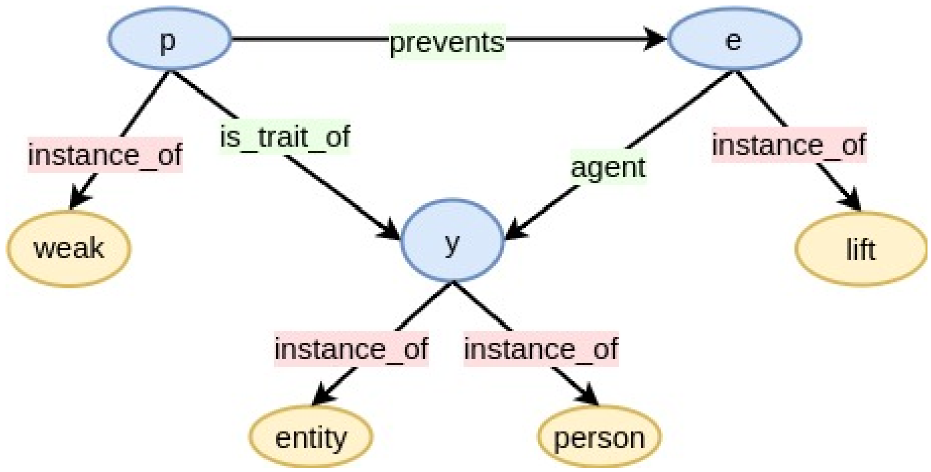

Figure 1.8: Graphical Representation of a Knowledge Instance from Knowledge Type 1

Definition 10 (Constant Node in a Sentence ($s\_const(x)$)) *Let $\mathcal{G}_A$ be a sentence's representation such that $\mathcal{G}_A$ is of the form $\mathcal{G}_S$ (from definition 2). We say that x is a constant node in $\mathcal{G}_A$ iff x has an outgoing edge labeled "$instance\_of$" to another node i in $\mathcal{G}_A$ i.e.,*

$$s\_const(x) \iff ((x, instance\_of, i) \in \mathcal{G}_A)$$

For example, let us consider a sentence's representation as shown in Figure 1.1. Then, according to the above definition following are the constant nodes.

`s_const(weak_12)`

`s_const(he_9)`

`s_const(so_11)`

`s_const(lift_5)`

`s_const(son_7)`

`s_const(man_2)`

`s_const(his_6)`

Definition 11 (Constant Node in a Question ($q\_const(x)$)) *Let $\mathcal{G}_A$ be a question representation such that $\mathcal{G}_A$ is of the form $\mathcal{G}_Q$ (from definition 3). We say that x is a constant* *node in $\mathcal{G}_A$ iff x has an outgoing edge labeled "instance_of" to another node i in $\mathcal{G}_A$ i.e.,*

$$q\_const(x) \iff ((x, instance\_of, i) \in \mathcal{G}_A)$$

For example, let us consider a question's representation as shown in Figure 1.2. Then, according to the above definition following are the constant nodes.

```
q_const(weak_3)
q_const(q_1)
```

Definition 12 (Constant Node in a Knowledge Instance ($k\_const(x)$)) *Let $\mathcal{G}_A$ be a required knowledge representation such that $\mathcal{G}_A$ is of the form $\mathcal{G}_\mathcal{K}$ (from definitions 4 to 9). We say that x is a constant node in $\mathcal{G}_A$ iff x has an outgoing edge labeled "instance_of" to another node i in $\mathcal{G}_A$ i.e.,*

$$k\_const(x) \iff ((x, instance\_of, i) \in \mathcal{G}_A)$$

For example, let us consider a required knowledge's representation as shown in Figure 1.8. Then, according to the above definition following are the constant nodes.

```
k_const(p)
k_const(e)
k_const(y)
```

Definition 13 (A Node Has-Parent in Sentence ($s\_has\_par(x)$)) *Let $\mathcal{G}_A$ be a sentence's representation such that $\mathcal{G}_A$ is of the form $\mathcal{G}_\mathcal{S}$ (from definition 2). We say that x has a parent node in $\mathcal{G}_A$ iff x is a constant node in $\mathcal{G}_A$ and x has an incoming edge from a constant node p in $\mathcal{G}_A$ i.e.,*

$$s\_has\_par(x) \iff ((p, r, x) \in \mathcal{G}_A) \wedge s\_const(x) \wedge s\_const(p)$$

For example, let us consider a sentence's representation as shown in Figure 1.1. Then, according to the above definition following nodes have parent(s).

```
s_has_par(he_9)
s_has_par(so_11)
s_has_par(lift_5)
s_has_par(son_7)
s_has_par(man_2)
s_has_par(his_6)
```

Definition 14 (A Node Has-Parent in Question ($q\_has\_par(x)$)) *Let $\mathcal{G}_A$ be a question's representation such that $\mathcal{G}_A$ is of the form $\mathcal{G}_Q$ (from definition 3). We say that x has a parent node in $\mathcal{G}_A$ iff x is a constant node in $\mathcal{G}_A$ and x has an incoming edge from a constant node p in $\mathcal{G}_A$ i.e.,*

$$q\_has\_par(x) \iff ((p,r,x) \in \mathcal{G}_A) \wedge q\_const(x) \wedge q\_const(p)$$

For example, let us consider a question's representation as shown in Figure 1.2. Then, according to the above definition following nodes have parent(s).

```
q_has_par(q_1)
```

Definition 15 (A Node Has-Parent in Domain Knowledge ($k\_has\_par(x)$)) *Let $\mathcal{G}_A$ be a required knowledge representation such that $\mathcal{G}_A$ is of the form $\mathcal{G}_K$ (from definitions 4 to 9). We say that x has a parent node in $\mathcal{G}_A$ iff x is a constant node in $\mathcal{G}_A$ and x has an incoming edge from a constant node p in $\mathcal{G}_A$ i.e.,*

$$k\_has\_par(x) \iff ((p,r,x) \in \mathcal{G}_A) \wedge k\_const(x) \wedge k\_const(p)$$

For example, let us consider a required knowledge's representation as shown in Figure 1.8. Then, according to the above definition following nodes have parents.

`k_has_par(e)`

`k_has_par(y)`

Definition 16 (A Node Has-Child in Sentence ($s\_has\_child(x)$)) *Let $\mathcal{G}_A$ be a sentence's representation such that $\mathcal{G}_A$ is of the form $\mathcal{G}_S$ (from definition 2). We say that x has a child node in $\mathcal{G}_A$ iff x is a constant node in $\mathcal{G}_A$ and x has an outgoing edge to a constant node c in $\mathcal{G}_A$ i.e.,*

$$s\_has\_child(x) \iff ((x,r,c) \in \mathcal{G}_A) \wedge s\_const(x) \wedge s\_const(c)$$

For example, let us consider a sentence's representation as shown in Figure 1.1. Then, according to the above definition following nodes have children.

`s_has_child(weak_12)`

`s_has_child(lift_5)`

`s_has_child(son_7)`

Definition 17 (A Node Has-Child in Question ($q\_has\_child(x)$)) *Let $\mathcal{G}_A$ be a question's representation such that $\mathcal{G}_A$ is of the form $\mathcal{G}_Q$ (from definition 3). We say that x has a child node in $\mathcal{G}_A$ iff x is a constant node in $\mathcal{G}_A$ and x has an outgoing edge to a constant node c in $\mathcal{G}_A$ i.e.,*

$$q\_has\_child(x) \iff ((x,r,c) \in \mathcal{G}_A) \wedge q\_const(x) \wedge q\_const(c)$$

For example, let us consider a question's representation as shown in Figure 1.2. Then, according to the above definition following nodes have children.

`q_has_child(weak_3)`

Definition 18 (A Node Has-Child in Domain Knowledge ($s\_has\_child(x)$)) *Let $\mathcal{G}_A$ be a required knowledge representation such that $\mathcal{G}_A$ is of the form $\mathcal{G}_\mathcal{K}$ (from definitions 4 to 9). We say that x has a child node in $\mathcal{G}_A$ iff x is a constant node in $\mathcal{G}_A$ and x has an outgoing edge to a constant node c in $\mathcal{G}_A$ i.e.,*

$$k\_has\_child(x) \iff ((x,r,c) \in \mathcal{G}_A) \wedge k\_const(x) \wedge k\_const(c)$$

For example, let us consider a required knowledge's representation as shown in Figure 1.8. Then, according to the above definition following nodes have children.

$k\_has\_child(p)$

$k\_has\_child(e)$

Definition 19 (Cross-Domain Sibling from Domain Knowledge to Sentence ($k\_s\_crossdom\_sib(x,y)$)) *Let $\mathcal{G}_A$ be a required knowledge's representation (i.e.,$\mathcal{G}_A$ is of the form $\mathcal{G}_\mathcal{K}$, see definitions 4 to 9) and $\mathcal{G}_B$ be a sentence's representation (i.e., $\mathcal{G}_B$ is of the form $\mathcal{G}_\mathcal{S}$, see definition 2). We say that a node y in $\mathcal{G}_B$ is a cross-required sibling of a node x in $\mathcal{G}_A$ (from required knowledge to sentence) iff all of the conditions below are satisfied.*

(a) *x is a constant node in $\mathcal{G}_A$,*

(b) *y is a constant node in $\mathcal{G}_B$,*

(c) *if there exists an outgoing edge labeled "$instance\_of$" from the node x to a node i in $\mathcal{G}_A$, then there also exists an outgoing edge labeled "$instance\_of$" from y to a node i in $\mathcal{G}_B$*

Logically,

$$k\_s\_crossdom\_sib(x,y) \iff (k\_const(x) \wedge s\_const(y) \wedge ((x, instance\_of, i) \in \mathcal{G}_A) \implies ((y, instance\_of, i) \in \mathcal{G}_B))$$

For example, let us consider a required knowledge's representation as shown in Figure 1.8 and a sentence's representation as shown in Figure 1.1. Then, according to the above definition following nodes are cross-domain siblings.

```
k_s_crossdom_sib(p,weak_12)
k_s_crossdom_sib(y,he_9)
k_s_crossdom_sib(y,son_7)
k_s_crossdom_sib(y,man_2)
k_s_crossdom_sib(y,his_6)
k_s_crossdom_sib(e,lift_5)
```

Definition 20 (Cross-Domain Clone from Domain Knowledge to Sentence ($k\_s\_crossdom\_clone(x,y)$**))** *Let $\mathcal{G}_A$ be a required knowledge's representation (i.e., $\mathcal{G}_A$ is of the form $\mathcal{G}_{\mathcal{K}}$, see definitions 4 to 9) and $\mathcal{G}_B$ be a sentence's representation (i.e., $\mathcal{G}_B$ is of the form $\mathcal{G}_{\mathcal{S}}$, see definition 2). We say that a node y in $\mathcal{G}_B$ is a cross-domain clone of a node x in $\mathcal{G}_A$ (from required knowledge to sentence) iff all of the following conditions are satisfied.*

(a) x is a constant node in $\mathcal{G}_A$,

(b) y is a constant node in $\mathcal{G}_B$,

(c) y is a cross-domain sibling of x from required knowledge to sentence (i.e., $k\_s\_crossdom\_sib(x,y)$),

(d) If (p_j, r_j, x) is an edge in $\mathcal{G}_A$, then (p'_j, r_j, y) is an edge in $\mathcal{G}_B$ such that $k\_s\_crossdom\_clone(p_j, p'_j)$, where p_j is a constant node in $\mathcal{G}_A$, and

(e) If (x, r_k, c_k) is an edge in $\mathcal{G}_A$, then (y, r_k, c'_k) is an edge in $\mathcal{G}_B$ such that $k\_s\_crossdom\_sib(c_k, c'_k)$, where c_k is a constant node in $\mathcal{G}_A$.

For example, let us consider a required knowledge's representation as shown in Figure 1.8 and a sentence's representation as shown in Figure 1.1. Then, according to the above definition following nodes are cross-domain clones.

```
k_s_crossdom_clone(p,weak_12)
k_s_crossdom_clone(e,lift_5)
k_s_crossdom_clone(y,man_2)
k_s_crossdom_clone(y,he_9)
```

Definition 21 (Merged Representation ($\mathcal{G}_M$)) *Let $\mathcal{G}_A$ be a sentence's representation (i.e., $\mathcal{G}_A$ is of the form $\mathcal{G}_S$, see definition 2) and $\mathcal{G}_B$ be a required knowledge representation (i.e., $\mathcal{G}_B$ is of the form $\mathcal{G}_K$, see definitions 4 to 9). A merged representation $\mathcal{G}_M$, is obtained by performing the following set of operations on $\mathcal{G}_A$ and $\mathcal{G}_B$.*

(i) copy all the nodes and edges of $\mathcal{G}_A$ into $\mathcal{G}_M$

(ii) if all the constant nodes in $\mathcal{G}_B$ have at-least one cross-domain clone in $\mathcal{G}_A$ then, for each constant node v_i in $\mathcal{G}_B$, for every two distinct constant nodes vs_j and vs_k in $\mathcal{G}_A$ such that they are cross-domain clones of v_i (i.e., $k\_s\_crossdom\_clone(v_i, vs_j)$, $k\_s\_crossdom\_clone(v_i, vs_k)$, and $vs_j \neq vs_k$), do the following if the respective conditions are satisfied.

(a) add (p_j, r_j, vs_k) to $\mathcal{G}_M$ if $(p_j, r_j, vs_j) \in \mathcal{G}_A$

(b) add (vs_k, r_j, c_j) to $\mathcal{G}_M$ if $(vs_j, r_j, c_j) \in \mathcal{G}_A$

For example, let us consider a required knowledge's representation as shown in Figure 1.8 and a sentence's representation as shown in Figure 1.1. Then, according to the above definition the merged representation is as shown in Figure 1.9.

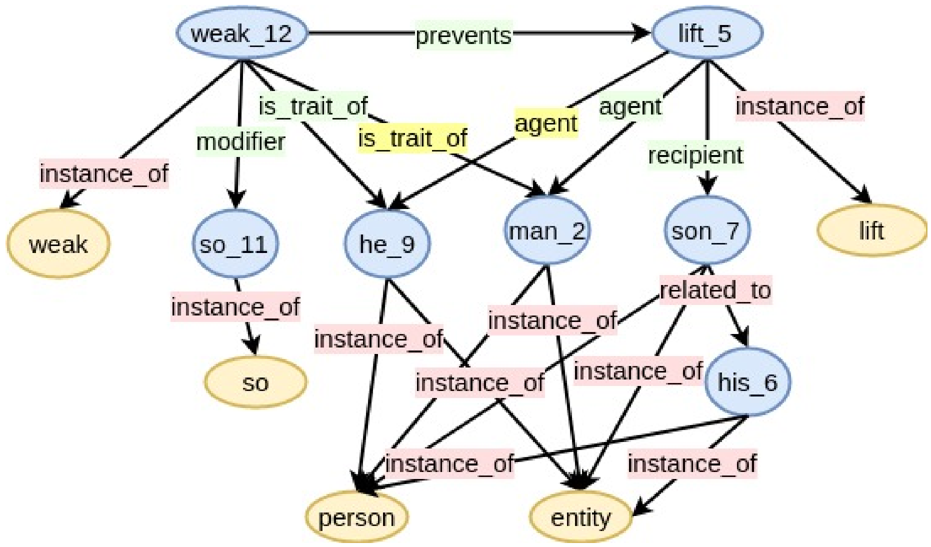

Figure 1.9: An Example of a Merged Graphical Representation

Definition 22 (Constant Node in Merged Representation ($m\_const(x)$)) *Let $\mathcal{G}_M$ be a merged representation from definition 21. We say that x is a constant node in $\mathcal{G}_M$ iff x has an outgoing edge labeled "$instance\_of$" to another node i in $\mathcal{G}_M$ i.e.,*

$$m\_const(x) \iff ((x, instance\_of, i) \in \mathcal{G}_M)$$

For example, let us consider the merged representation as shown in Figure 1.9. Then, according to the above definition following are the constant nodes.

m_const(he_9)

m_const(so_11)

m_const(weak_12)

m_const(lift_5)

m_const(son_7)

m_const(man_2)

m_const(his_6)

Definition 23 (A Node Has-Parent in Merged Representation ($m\_has\_par(x)$)) *Let $\mathcal{G}_M$ be a merged representation from definition 21. We say that x has a parent node in $\mathcal{G}_M$ iff x is a constant node in $\mathcal{G}_M$ and x has an incoming edge from a constant node p in $\mathcal{G}_M$ i.e.,*

$$m\_has\_par(x) \iff ((p,r,x) \in \mathcal{G}_M) \wedge m\_const(x) \wedge m\_const(p)$$

For example, let us consider the merged representation as shown in Figure 1.9. Then, according to the above definition following nodes have parent(s).

m_has_par(he_9)

m_has_par(so_11)

m_has_par(man_2)

m_has_par(son_7)

m_has_par(his_6)

m_has_par(lift_5)

Definition 24 (A Node Has-Child in Merged Representation ($m\_has\_child(x)$)) *Let $\mathcal{G}_M$ be a merged representation from definition 21. We say that x has a child node in $\mathcal{G}_M$ iff x is a constant node in $\mathcal{G}_M$ and x has an outgoing edge to a constant node c in $\mathcal{G}_M$ i.e.,*

$$m\_has\_child(x) \iff ((x,r,c) \in \mathcal{G}_M) \wedge m\_const(x) \wedge m\_const(c)$$

For example, let us consider the merged representation as shown in Figure 1.9. Then, according to the above definition following nodes have children.

m_has_child(weak_12)

m_has_child(lift_5)

m_has_child(son_7)

Definition 25 (Cross-Domain Sibling from Question to Merged Representation) *Let $\mathcal{G}_A$ be a question's representation (i.e., $\mathcal{G}_A$ is of the form $\mathcal{G}_Q$, see definition 3) and $\mathcal{G}_M$ is the* *merged representation from definition 21. We say that a node x in $\mathcal{G}_A$ has a cross-domain sibling node y in $\mathcal{G}_M$ (from question to merged representation) iff all of the conditioned mentioned below are satisfied.*

(a) x is a constant node in $\mathcal{G}_A$,

(b) y is a constant node in $\mathcal{G}_M$,

(c) if there exists an outgoing edge labeled "$instance\_of$" from the node x to a node i in $\mathcal{G}_A$ such that i is not "q" then there exists an outgoing edge labeled "$instance\_of$" from y to a node i in $\mathcal{G}_M$.

Logically,

$$q\_m\_crossdom\_sib(x,y) \iff (q\_const(x) \wedge m\_const(y) \wedge ((x, instance\_of, i) \in \mathcal{G}_A \wedge i \neq q) \implies ((y, instance\_of, i) \in \mathcal{G}_M))$$

For example, let us consider the merged representation as shown in Figure 1.9 and a question's representation as shown in Figure 1.2. Then, according to the above definition following nodes are cross-domain siblings.

```
q_m_crossdom_sib(q_1,he_9)
q_m_crossdom_sib(q_1,son_7)
q_m_crossdom_sib(q_1,man_2)
q_m_crossdom_sib(q_1,his_6)
q_m_crossdom_sib(weak_3,weak_12)
```

Definition 26 (Cross-Domain Clone from Question to Merged Representation) *Let $\mathcal{G}_A$ be a question's representation (i.e., $\mathcal{G}_A$ is of the form $\mathcal{G}_Q$, see definition 3) and $\mathcal{G}_M$ is the merged representation from definition 21. We say that a node y in $\mathcal{G}_M$ is a cross-domain clone of node x in $\mathcal{G}_A$ (from question to the merged representation) iff all of the following conditions are satisfied.*

(a) *x is a constant node in $\mathcal{G}_A$,*

(b) *y is a constant node in $\mathcal{G}_M$,*

(c) *y is a cross-domain sibling of x from the question to merged representation (i.e., $q\_m\_crossdom\_sib(x,y)$),*

(d) *If (p_j, r_j, x) is an edge in $\mathcal{G}_A$, then (p'_j, r_j, y) is an edge in $\mathcal{G}_M$ such that $q\_m\_crossdom\_clone(p_j, p'_j)$, where p_j is a constant node in $\mathcal{G}_A$, and*

(e) *If (x, r_k, c_k) is an edge in $\mathcal{G}_A$, then (y, r_k, c'_k) is an edge in $\mathcal{G}_M$ such that $q\_m\_crossdom\_sib(c_k, c'_k)$, where c_k is a constant node in $\mathcal{G}_A$.*

For example, let us consider the merged representation as shown in Figure 1.9 and a question's representation as shown in Figure 1.2. Then, according to the above definition following nodes are cross-domain clones.

```
q_m_crossdom_clone(weak_3,weak_12)
q_m_crossdom_clone(q_1,he_9)
q_m_crossdom_clone(q_1,man_2)
```

1.2.3 Defining Answer Deduction Phase of the Reasoning Framework

Definition 27 (Final Answers ($ans(q_i,x)$)) *Let $\mathcal{G}_A$ be a question's representation (i.e., $\mathcal{G}_A$ is of the form $\mathcal{G}_Q$, see definition 3) and $\mathcal{G}_M$ be a merged representation from definition 21. We say that if each constant node in $\mathcal{G}_Q$ has a cross-domain clone in $\mathcal{G}_M$, q_i is an unknown node in $\mathcal{G}_Q$ (i.e. $(q_i, instance\_of, q) \in \mathcal{G}_A$), and x is a cross-domain clone of q_i from $\mathcal{G}_A$ to $\mathcal{G}_M$ (i.e., $q\_m\_crossdom\_clone(q_i,x)$), then x is an answer to the unknown node, q_i, in the input question. Logically,*

$$ans(q_i,x) \iff q\_m\_crossdom\_clone(q_i,x) \wedge ((q_i, instance\_of, q) \in \mathcal{G}_A) \wedge q\_all\_covered$$

where, q_all_covered is true if and only if each constant node in $\mathcal{G}_Q$ has at-least one clone in $\mathcal{G}_M$.

For example, let us consider a sentence's representation as shown in Figure 1.1, a question's representation as shown in Figure 1.2, a required knowledge's representation as shown in Figure 1.8 and the merged representation as shown in Figure 1.9. Then, according to the above definition following are the answers to the question.

```
ans(q_1,he_9)
ans(q_1,man_2)
```

1.3 Answer Set Programming Encoding of the Reasoning Aspects

We used a logic language called Answer Set Programming as the implementation language of our reasoning framework. Below is the formal definition of the language.

1.3.1 Answer Set Programming

We chose Answer Set Programming (ASP) (Gelfond and Lifschitz, 1988b) as our knowledge representation language because of the following reasons:

1. ASP has simple syntax yet is expressive; it is non-monotonic,

2. ASP has a strong theoretical foundation with many building-block results (Baral, 2003), allowing us to prove the correctness of our implementation in this work.

3. It has several efficient solvers: ('Leone *et al.*, 2006)

An ASP program is a collection of rules of the form:

$$a \leftarrow a_1, \ldots, a_m, \textit{not } a_{m+1}, \ldots, \textit{not } a_{m+n}$$

where a, $a_1,...,a_{m+n}$ are atoms. The rule reads as "a is true if $a_1...a_m$ are all known to be true and $a_{m+1}...a_{m+n}$ can be assumed to be false". The semantics of answer set programs are defined using answer sets. An entailment relation ($\models$) with respect to answer set programs is defined as follows: a program Π entails an atom p iff p is true in all the answer sets of Π.

In this section we present the ASP encoding of the inputs (the sentence, the question and the required knowledge) and the rules (based on the definitions 2-27) that are used to perform the commonsense reasoning on the inputs.

1.3.2 ASP Encoding of the Inputs

There are three inputs to the reasoning framework. All three inputs, namely, the sentence, the question and the corresponding required knowledge are formally represented as rooted, edge labeled directed acyclic graphs. These graphs are easily translated into RDF style triples [4] of the form *(h,l,t)*, where *h* and *t* represent two nodes in the graph and *l* represents the label of a directed edge from *h* to *t*. To differentiate between inputs, we appended the triple with different identifiers for each input. For example an edge in a sentence is represented as $has\_s(h,l,t)$, and edge in a question is represented as $has\_q(h,l,t)$ and an edge in a knowledge instance is represented as $has\_k(h,l,t)$. The RDF style triple representations of a set of actual inputs from the WSC problem are as shown below.

```
% Sentence: The man could not lift his son because he was
   so weak .
% Sentence's ASP encoding:
has_s(he_9,instance_of,person).
has_s(he_9,instance_of,entity).
```

[4] https://www.w3.org/TR/2004/REC-rdf-concepts-20040210/#section-triples

```
has_s(weak_12,is_trait_of,he_9).
has_s(weak_12,modifier,so_11).
has_s(so_11,instance_of,so).
has_s(weak_12,instance_of,weak).
has_s(lift_5,instance_of,lift).
has_s(lift_5,agent,man_2).
has_s(lift_5,recipient,son_7).
has_s(son_7,instance_of,person).
has_s(son_7,instance_of,entity).
has_s(man_2,instance_of,person).
has_s(man_2,instance_of,entity).
has_s(son_7,related_to,his_6).
has_s(his_6,instance_of,entity).
has_s(his_6,instance_of,person).
has_s(weak_12,prevents,lift_5).

% Question: Who was weak ?
% Question's ASP encoding:
has_q(weak_3,is_trait_of,q_1).
has_q(q_1,instance_of,q).
has_q(q_1,instance_of,person).
has_q(q_1,instance_of,entity).
has_q(weak_3,instance_of,weak).

% Knowledge Instance: x is weak may prevent x from lifting


% Knowledge Instance's ASP encoding:
```

```
has_k(p,instance_of,weak).
has_k(p,is_trait_of,y).
has_k(y,instance_of,entity).
has_k(y,instance_of,person).
has_k(e,instance_of,lift).
has_k(e,agent,y).
has_k(p,prevents,e).
```

1.3.3 ASP Encoding of the Intermediate Operations

ASP Rule to Obtain Constant Nodes in Sentence's Representation

According to the definition 10 ($s\_const(x)$), a node x in a sentence's representation (say $\mathcal{G}_A$) is a constant node iff x has an outgoing edge labeled "$instance\_of$". The following ASP rule is used to encode definition 10.

```
n1: s_const(X) :- has_s(X,instance_of,I).
```

Here, `s_const(X)` represents that `X` is a constant node in the given sentence's representation. The output of the rule, based on the inputs mentioned in the section 1.3.2 is shown below.

```
s_const(weak_12)
s_const(he_9)
s_const(so_11)
s_const(lift_5)
s_const(son_7)
s_const(man_2)
s_const(his_6)
```

ASP Rule to Obtain Constant Nodes in Question's Representation

According to the definition 11 ($q\_const(x)$), a node x in a question's representation (say $\mathcal{G}_A$) is a constant node iff x has an outgoing edge labeled "$instance\_of$". Following ASP rule is used to encode definition 11.

```
n2: q_const(X) :- has_q(X,instance_of,I).
```

Here, `q_const(X)` represents that `X` is a constant node in the given question's representation. The output of the rule, based on the input mentioned in the section 1.3.2 is as shown below.

q_const(weak_3)

q_const(q_1)

ASP Rule to Obtain Constant Nodes in Domain Knowledge's Representation

According to the definition 12 ($k\_const(x)$), a node x in a required knowledge's representation (say $\mathcal{G}_A$) is a constant node iff x has an outgoing edge labeled "$instance\_of$". Following ASP rule is used to encode definition 12.

```
n3: k_const(X) :- has_k(X,instance_of,I).
```

Here, `k_const(X)` represents that `X` is a constant node in the given required knowledge's representation. The output of the rule, based on the input mentioned in the section 1.3.2 is as shown below.

k_const(p)

k_const(e)

k_const(y)

ASP Rule to Obtain Has-Parent Information in Sentence's Representation

According to the definition 13 ($s\_has\_par(x)$), a node x in a sentence's representation, say $\mathcal{G}_A$, has a parent in $\mathcal{G}_A$ iff x is a constant node and x has an incoming edge from a constant node in $\mathcal{G}_A$. Following ASP rule is used to encode the definition 13.

```
p1: s_has_par(X) :- has_s(P,R,X), s_const(X), s_const(P).
```

Here, `s_has_par(X)` represents that `X` has a parent in the representation of the input sentence. The output of the rule, based on the inputs mentioned in the section 1.3.2 and the outputs of the ASP rules in sections 1.3.3 to 1.3.3 is as shown below.

s_has_par(he_9)

s_has_par(so_11)

s_has_par(lift_5)

s_has_par(son_7)

s_has_par(man_2)

s_has_par(his_6)

ASP Rule to Obtain Has-Parent Information in Question's Representation

According to the definition 14 ($s\_has\_par(x)$), a node x in a question's representation, say $\mathcal{G}_A$, has a parent in $\mathcal{G}_A$ iff x is a constant node and x has an incoming edge from a constant node in $\mathcal{G}_A$. Following ASP rule is used to encode definition 13.

```
p2: q_has_par(X) :- has_q(P,R,X), q_const(X), q_const(P).
```

Here, `s_has_par(X)` represents that `X` has a parent in the representation of the input sentence. The output of the rule, based on the inputs mentioned in the section 1.3.2 is as shown below.

q_has_par(q_1)

ASP Rule to Obtain Has-Parent Information in Domain Knowledge Representation

According to the definition of $k\_has\_par(x)$ above, a constant node in the required knowledge representation $\mathcal{G}_A$ has a parent in $\mathcal{G}_A$ if it has an incoming edge from a constant node in $\mathcal{G}_A$. Following ASP rule is used to extract this relationship from $\mathcal{G}_A$.

```
p3: k_has_par(X) :- has_k(P,R,X), k_const(X), k_const(P).
```

where, `k_has_par(X)` represents that `X` has a parent in the representation of the input sentence. The output of the rule, based on the input mentioned above is shown below.

k_has_par(e)

k_has_par(y)

ASP Rule to Obtain Has-Child Information in Sentence's Representation

According to the definition of $s\_has\_child(x)$ above, a constant node in the textual representation $\mathcal{G}_A$ has a child in $\mathcal{G}_A$ if it has an outgoing edge to a constant node in $\mathcal{G}_A$. Following ASP rule is used to extract this relationship from $\mathcal{G}_A$.

```
h1: s_has_child(X) :- has_s(X,R,C), s_const(X), s_const(C)
     .
```

where, `s_has_child(X)` represents that `X` has a child in the representation of the input sentence. The output of the rule, based on the input mentioned above is shown below.

s_has_child(weak_12)

s_has_child(lift_5)

s_has_child(son_7)

ASP Rule to Obtain Has-Child Information in Question's Representation

According to the definition of $q\_has\_child(x)$ above, a constant node in the question representation $\mathcal{G}_A$ has a child in $\mathcal{G}_A$ if it has an outgoing edge to a constant node in $\mathcal{G}_A$. Following

ASP rule is used to extract this relationship from $\mathcal{G}_A$.

```
h2: q_has_child(X) :- has_q(X,R,C), q_const(X), q_const(C)
    .
```

where, `q_has_child(X)` represents that `X` has a child in the representation of the input sentence. The output of the rule, based on the input mentioned above is shown below.

```
q_has_child(weak_3)
```

ASP Rule to Obtain Has-Child Information in Domain Knowledge Representation

According to the definition of $k\_has\_child(x)$ above, a constant node in the required knowledge's representation $\mathcal{G}_A$ has a child in $\mathcal{G}_A$ if it has an outgoing edge to a constant node in $\mathcal{G}_A$. Following ASP rule is used to extract this relationship from $\mathcal{G}_A$.

```
h3: k_has_child(X) :- has_k(X,R,C), k_const(X), k_const(C)
    .
```

where, `k_has_child(X)` represents that `X` has a child in the representation of the input sentence. The output of the rule, based on the input mentioned above is shown below.

```
k_has_child(p)
k_has_child(e)
```

ASP Rules to Obtain Cross-Domain Siblings from Domain Knowledge to Sentence

According to the definition of $k\_s\_crossdom\_sib(x,y)$ above, a constant node x in the required knowledge's representation $\mathcal{G}_A$ has a cross-domain sibling node y in the textual representation $\mathcal{G}_B$ if all the conceptual classes of x are also the conceptual classes of y. Following ASP rules are used to extract this relationship.

```
s1: not_k_s_crossdom_sib(X,Y) :- has_k(X,instance_of,I1),
                                  has_k(X,instance_of,I2),
```

```
                          has_s(Y,instance_of,I1),
                          not has_s(Y,instance_of,I2),
                          I1!=I2.

s2: not_k_s_crossdom_sib(X,Y) :- has_k(X,instance_of,I1),
                          has_k(X,instance_of,I2),
                          not has_s(Y,instance_of,I1),
                          has_s(Y,instance_of,I2),
                          I1!=I2.

s3: not_k_s_crossdom_sib(X,Y) :- has_k(X,instance_of,I1),
                          has_k(X,instance_of,I2),
                          not has_s(Y,instance_of,I1),
                          not has_s(Y,instance_of,I2),
                          s_const(Y), I1!=I2.

s4: k_s_crossdom_sib(X,Y) :- has_s(Y,instance_of,I),
                         has_k(X,instance_of,I),
                         not not_k_s_crossdom_sib(X,Y),
                         s_const(Y),
                         k_const(X).
```

where, `k_s_crossdom_sib(X,Y)` represents that `X` in the representation of required knowledge has a cross-domain sibling `Y` in the textual representation. The output of the rule, based on the input mentioned above is shown below.

k_s_crossdom_sib(p,weak_12)

k_s_crossdom_sib(y,he_9)

k_s_crossdom_sib(y,son_7)

k_s_crossdom_sib(y,man_2)

k_s_crossdom_sib(y,his_6)

k_s_crossdom_sib(e,lift_5)

ASP Rules to Obtain Cross-Domain Clones from Domain Knowledge to Sentence

According to the definition of *k_s_crossdom_clone*(x,y) above, a node x in the required knowledge's representation $\mathcal{G}_A$ has a cross-domain clone node y in the textual representation $\mathcal{G}_B$ if they are cross-domain siblings, their parents are cross-domain clones (in case there are any) and their children are cross-domain siblings (in case there are any). Following ASP rules are used to extract this relationship.

```
c1: k_s_crossdom_clone(X,Y) :- k_s_crossdom_sib(X,Y),
                               has_k(Pj,Rj,X),
                               has_s(Pj_prime,Rj,Y),
                               k_s_crossdom_clone(Pj,Pj_prime),
                               k_const(Pj),
                               has_k(X,Rk,Cj),
                               has_s(Y,Rk,Cj_prime),
                               k_s_crossdom_sib(Cj,Cj_prime),
                               k_const(Cj).


c2: k_s_crossdom_clone(X,Y) :- k_s_crossdom_sib(X,Y),
                               not k_has_par(X),
                               has_k(X,Rk,Cj),
                               has_s(Y,Rk,Cj_prime),
```

```
                              k_s_crossdom_sib(Cj,Cj_prime),
                              k_const(Cj).

c3: k_s_crossdom_clone(X,Y) :- k_s_crossdom_sib(X,Y),
                              has_k(Pj,Rj,X),
                              has_s(Pj_prime,Rj,Y),
                              k_s_crossdom_clone(Pj,Pj_prime),
                              k_const(Pj),
                              not k_has_child(X).

c4: k_s_crossdom_clone(X,Y) :- k_s_crossdom_sib(X,Y),
                              not k_has_par(X),
                              not k_has_child(X).
```

where, `k_s_crossdom_clone(X,Y)` represents that `X` in the representation of required knowledge has a cross-domain clone `Y` in the textual representation. The output of these rules, based on the input mentioned above is shown below.

```
k_s_crossdom_clone(p,weak_12)
k_s_crossdom_clone(e,lift_5)
k_s_crossdom_clone(y,man_2)
k_s_crossdom_clone(y,he_9)
```

ASP Rules to Obtain the Merged Representation

Intuitively, a merged representation of a sentence and knowledge instance represents a representation that contains all the information provided in the sentence and also the knowledge contained in the knowledge instance.

According to the definition of a merged representation (Definition 21), a merged representation of an input sentence and a knowledge instance is a representation that contains all the information provided in the sentence and also the knowledge contained in the knowledge instance. It is generated in two parts.

Firstly, all the nodes and edges from the sentence's representation are copied as is to the merged representation. An ASP rule to accomplish that is as shown below.

```
m1: has_m(X,R,Y) :- has_s(X,R,Y)
```

Secondly, if each the constant node in the knowledge instance's representation has a cross-domain clone in the sentence's representation then for each constant node in the knowledge instance's representation, all of its cross-domain clones in the sentence's representation are merged as one node. This part is encoded in ASP in two steps. Step 1 involves the rules to identify if each constant node in the knowledge instance's representation has a cross-domain clone in the sentence's representation. Following are the rules.

```
m2: k_covered(X) :- k_const(X),
                    s_const(Y),
                    k_s_crossdom_clone(X,Y).


m3: k_not_all_covered :- not k_covered(X),
                         k_const(X).


m4: k_all_covered :- not k_not_all_covered.
```

Step 2 involves the rule to merge cross-domain clone nodes. Following are the rules.

```
m5: has_m(X,R,Y) :- has_s(X,R,Y1),
                    has_s(X2,R2,Y),
                    Y1!=Y,
                    k_s_crossdom_clone(Y_k,Y1),
```

```
                    k_s_crossdom_clone(Y_k,Y),
                    k_all_covered.

m6: has_m(X,R,Y) :- has_s(X1,R,Y),
                    has_s(X,R2,Y2),
                    X1!=X,
                    k_s_crossdom_clone(X_k,X1),
                    k_s_crossdom_clone(X_k,X),
                    k_all_covered.
```

where, has_m(X,R,Y) represents a directed edge, labeled R, from the node X to the node Y in the merged representation $\mathcal{G}_M$.

The output of these rules, based on the input mentioned above is shown below.

```
has_m(weak_12,is_trait_of,he_9)
has_m(weak_12,modifier,so_11)
has_m(lift_5,agent,man_2)
has_m(lift_5,recipient,son_7)
has_m(son_7,related_to,his_6)
has_m(lift_5,agent,he_9)
has_m(weak_12,is_trait_of,man_2)
has_m(weak_12,prevents,lift_5)
has_m(he_9,instance_of,person)
has_m(he_9,instance_of,entity)
has_m(son_7,instance_of,person)
has_m(son_7,instance_of,entity)
has_m(man_2,instance_of,person)
```

has_m(man_2,instance_of,entity)

has_m(his_6,instance_of,person)

has_m(his_6,instance_of,entity)

has_m(so_11,instance_of,so)

has_m(weak_12,instance_of,weak)

has_m(lift_5,instance_of,lift)

ASP Rule to Obtain Constant Nodes in Merged Representation

According to the definition of $m\_const(x)$ above, a node x in the required knowledge representation $\mathcal{G}_A$ is a constant node if it has an outgoing edge labeled "*instance_of*". Following ASP rule is used to extract this relationship from $\mathcal{G}_A$.

```
n4: m_const(X) :- has_m(X,instance_of,I).
```

where, `m_const(X)` represents that `X` is a constant node in the required knowledge representation. The output of the rule, based on the input mentioned above is shown below.

m_const(he_9)

m_const(so_11)

m_const(weak_12)

m_const(lift_5)

m_const(son_7)

m_const(man_2)

m_const(his_6)

ASP Rule to Obtain Has-Parent Information in Merged Representation

According to the definition of $m\_has\_par(x)$ above, a constant node in the merged representation $\mathcal{G}_A$ has a parent in $\mathcal{G}_A$ if it has an incoming edge from a constant node in $\mathcal{G}_A$. Following ASP rule are used to extract this relationship from $\mathcal{G}_A$.

```
p4: m_has_par(X) :- has_m(P,R,X), m_const(X), m_const(P).
```

where, `m_has_par(X)` represents that `X` has a parent in the representation of the input sentence. The output of the rule, based on the input mentioned above is shown below.

```
m_has_par(he_9)
m_has_par(so_11)
m_has_par(man_2)
m_has_par(son_7)
m_has_par(his_6)
m_has_par(lift_5)
```

ASP Rule to Obtain Has-Child Information in Merged Representation

According to the definition of $m\_has\_child(x)$ above, a constant node in the merged representation $\mathcal{G}_A$ has a child in $\mathcal{G}_A$ if it has an outgoing edge to a constant node in $\mathcal{G}_A$. Following ASP rule is used to extract this relationship from $\mathcal{G}_A$.

```
h4: m_has_child(X) :- has_m(X,R,C), m_const(X), m_const(C)
    .
```

where, `m_has_child(X)` represents that `X` has a child in the representation of the input sentence. The output of the rule, based on the input mentioned above is shown below.

```
m_has_child(weak_12)
m_has_child(lift_5)
m_has_child(son_7)
```

ASP Rules to Obtain Cross-Domain Siblings from Question to Merged Representation

According to the definition of $q\_m\_crossdom\_sib(x,y)$ above, a constant node x in the question's representation $\mathcal{G}_A$ has a cross-domain sibling node y in the merged representation $\mathcal{G}_B$

if all the conceptual classes of x are also the conceptual classes of y. Following ASP rules are used to extract this relationship.

```
s5: not_q_m_crossdom_sib(X,Y) :- has_q(X,instance_of,I1),
                                 has_q(X,instance_of,I2),
                                 has_m(Y,instance_of,I1),
                                 not has_m(Y,instance_of,I2),
                                 I1!=I2, I1!=q, I2!=q.

s6: not_q_m_crossdom_sib(X,Y) :- has_q(X,instance_of,I1),
                                 has_q(X,instance_of,I2),
                                 not has_m(Y,instance_of,I1),
                                 has_m(Y,instance_of,I2),
                                 I1!=I2, I1!=q, I2!=q.

s7: not_q_m_crossdom_sib(X,Y) :- has_q(X,instance_of,I1),
                                 has_q(X,instance_of,I2),
                                 not has_m(Y,instance_of,I1),
                                 not has_m(Y,instance_of,I2),
                                 I1!=I2, I1!=q, I2!=q,
                                 m_const(Y).

s8: q_m_crossdom_sib(X,Y) :- has_m(Y,instance_of,I),
                            has_q(X,instance_of,I),
                            not not_q_m_crossdom_sib(X,Y),
                            m_const(Y),
                            q_const(X).
```

where, `q_m_crossdom_sib(X,Y)` represents that `X` in the representation of the question has a cross-domain sibling `Y` in the merged representation. The output of the rule, based on the input mentioned above is shown below.

```
q_m_crossdom_sib(q_1,he_9)
q_m_crossdom_sib(q_1,son_7)
q_m_crossdom_sib(q_1,man_2)
q_m_crossdom_sib(q_1,his_6)
q_m_crossdom_sib(weak_3,weak_12)
```

ASP Rules to Obtain Cross-Domain Clones from Question to Merged Representation

According to the definition of $q\_m\_crossdom\_clone(x,y)$ above, a node x in the question's representation $\mathcal{G}_A$ has a cross-domain clone node y in the merged representation $\mathcal{G}_B$ if they are cross-domain siblings, their parents are cross-domain clones (in case there are any) and their children are cross-domain siblings (in case there are any). Following ASP rules are used to extract this relationship.

```
c5: q_m_crossdom_clone(X,Y) :- q_m_crossdom_sib(X,Y),
                               has_q(Pj,Rj,X),
                               has_m(Pj_prime,Rj,Y),
                               q_m_crossdom_clone(Pj,Pj_prime),
                               q_const(Pj),
                               has_q(X,Rk,Cj),
                               has_m(Y,Rk,Cj_prime),
                               q_m_crossdom_sib(Cj,Cj_prime),
                               q_const(Cj).


c6: q_m_crossdom_clone(X,Y) :- q_m_crossdom_sib(X,Y),
```

```
                        not q_has_par(X),
                        has_q(X,Rk,Cj),
                        has_m(Y,Rk,Cj_prime),
                        q_m_crossdom_sib(Cj,Cj_prime),
                        q_const(Cj).
```

```
c7: q_m_crossdom_clone(X,Y) :- q_m_crossdom_sib(X,Y),
                        has_q(Pj,Rj,X),
                        has_m(Pj_prime,Rj,Y),
                        q_m_crossdom_clone(Pj,Pj_prime),
                        q_const(Pj),
                        not q_has_child(X).
```

```
c8: q_m_crossdom_clone(X,Y) :- q_m_crossdom_sib(X,Y),
                        not q_has_par(X),
                        not q_has_child(X).
```

where, `q_m_crossdom_clone(X,Y)` represents that `X` in the representation of the question has a cross-domain clone `Y` in the merged representation. The output of these rules, based on the input mentioned above is shown below.

q_m_crossdom_clone(weak_3,weak_12)

q_m_crossdom_clone(q_1,he_9)

q_m_crossdom_clone(q_1,man_2)

ASP Rules to Obtain the Final Answers

According to the definition of $ans(q_i,x)$ above, the set of answers, with respect to an unknown node in the question, consists of the cross-domain clone nodes of the unknown node from the formal representation of the input question to the merged representation. The definition is encoded in ASP in two steps. Step 1 involves the rules to identify if each constant node in a question's representation has a cross-domain clone in the merged representation. Following are the rules.

```
a1: q_covered(X) :- q_const(X),
                    m_const(Y),
                    q_m_crossdom_clone(X,Y).


a2: q_not_all_covered :- not q_covered(X),
                         q_const(X).


a3: q_all_covered :- not q_not_all_covered.
```

Step 2 involves the answer deduction rule mentioned below.

```
a4: ans(Q,X) :- crossdom_clone(q,m,Q,X),
            has_q(Q,instance_of,q),
            q_all_covered.
```

where `ans(Q,X)` represents that the answer to the question, with respect to the unknown entity `Q`, is `X`. `q_m_crossdom_clone(Q,X)` represents that the unknown node, `Q`, in the question's representation has a cross-domain clone `X` in the merged representation, and `has_q(Q,instance_of,q)` represents that `Q` is an unknown node in the given question.

The output of this rule, based on the input mentioned above is shown below.

```
ans(q_1,he_9)

ans(q_1,man_2)
```

Theorems and Proofs

Definition 28 Revisited: *Given the graphical representations of a sentence ($\mathcal{G}_S$), a question ($\mathcal{G}_Q$) and the corresponding required knowledge ($\mathcal{G}_K$), the AnsProlog program $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$ is the answer set program consisting of the following:*

(i) *the facts of the form $has\_s(h,l,t)$, $has\_q(h,l,t)$, and $has\_k(h,l,t)$, representing the edges in $\mathcal{G}_S$, $\mathcal{G}_Q$ and $\mathcal{G}_K$ respectively.*

(ii) *the rules `n1` to `n4` for constant nodes, `p1` to `p4` for has-parent information, `h1` to `h4` for has-child information, `s1` to `s8` for cross-domain siblings, `c1` to `c8` for cross-domain clones, `m1` to `m8` for the merged representation, and `a1` to `a4` for final answer(s) to the given question.*

Proposition 1 *(Gelfond and Lifschitz, 1988a) Any stratified AnsProlog program is categorical and it has a unique answer set.*

Lemma 14 *1 Let $\mathcal{G}_S$, $\mathcal{G}_Q$, and $\mathcal{G}_K$ be a sentence's representation, a question's representation and the representation of a required knowledge instance respectively. Then $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$ is stratified.*

Proof 1 *According to the definition of a stratified AnsProlog program in (Baral, 2003), an AnsProlog program Π is stratified if there exists a stratification ($\pi_0, ..., \pi_k$) of Π.*
Also, a partition $\pi_0, ..., \pi_k$ of the set of all predicate symbols of Π is a stratification of Π, if for any rule of the type $A_0 \leftarrow A_1, ..., A_m, \textbf{\textit{not}} A_{m+1}, ..., \textbf{\textit{not}} A_n$, and for any $p \in \pi_s$, $0 \leq s \leq k$ if $A_0 \in atoms(p)$, then:

(a) for every $1 \leq i \leq m$ there is q and $j \leq s$ such that $q \in \pi_j$ and $A_i \in atoms(q)$, and

(b) for every $m+1 \leq i \leq n$ there is q and $j < s$ such that $q \in \pi_j$ and $A_i \in atoms(q)$

Now, let us divide all the predicates in $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$ in the following manner,
$\pi_0 = \{has\_s, has\_q, has\_k\}$,
$\pi_1 = \{s\_const, q\_const, k\_const\}$,
$\pi_2 = \{s\_has\_par, q\_has\_par, k\_has\_par, s\_has\_child, q\_has\_child, k\_has\_child\}$,
$\pi_3 = \{not\_k\_s\_crossdom\_sib\}$,
$\pi_4 = \{k\_s\_crossdom\_sib\}$,
$\pi_5 = \{k\_s\_crossdom\_clone\}$,
$\pi_6 = \{k\_covered\}$,
$\pi_7 = \{k\_not\_all\_covered\}$,
$\pi_8 = \{k\_all\_covered\}$,
$\pi_9 = \{has\_m\}$,
$\pi_{10} = \{m\_const\}$,
$\pi_{11} = \{m\_has\_par, m\_has\_child\}$,
$\pi_{12} = \{not\_q\_m\_crossdom\_sib\}$,
$\pi_{13} = \{q\_m\_crossdom\_sib\}$,

$\pi_{14} = \{q\_m\_crossdom\_clone\}$, *and*
$\pi_{15} = \{q\_covered\}$,
$\pi_{16} = \{q\_not\_all\_covered\}$,
$\pi_{17} = \{q\_all\_covered\}$,
$\pi_{18} = \{ans\}$

The above partitioning of all the predicates in $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$ satisfies the definition of a stratification of an AnsProlog program and therefore $\pi_0, ..., \pi_{10}$ is a stratification of $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$.

This means that there exists a stratification of $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$. Hence, $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$ is stratified.

Lemma 15 *2 Let $\mathcal{G}_S$, $\mathcal{G}_Q$, and $\mathcal{G}_K$ be a sentence's representation (by Definition 2), a question's representation (by Definition 3) and a representation of the corresponding required knowledge (by Definitions 4-9) respectively. Then $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$ has a unique answer set.*

Proof 2 *By Proposition 3 and Lemma 1, $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$ is categorical and it has a unique answer set.*

Proposition 2 ((Marek and Subrahmanian, 1989)) *(a)* ***Forced atom Proposition:*** *Let S be an answer set of an AnsProlog program Π. For any ground instance of a rule in Π of the type $A_0 \leftarrow A_1, ..., A_m, \textbf{not} A_{m+1}, ..., \textbf{not} A_n$. If $\{A_1, ..., A_m\} \subseteq S$ and $\{A_{m+1}, ..., A_n\} \cap S = \phi$ then $A_0 \in S$.*

(b) ***Supporting rule proposition:*** *If S is an answer set of an AnsProlog program Π then S is supported by Π. That is, if $A_0 \in S$, then there exists a ground instance of a rule in Π of the type $A_0 \leftarrow A_1, ..., A_m, \textbf{not} A_{m+1}, ..., \textbf{not} A_n$. Such that $\{A_1, ..., A_m\} \subseteq S$ and $\{A_{m+1}, ..., A_n\} \cap S = \phi$.*

Lemma 16 *Let $\mathcal{G}_S$, $\mathcal{G}_Q$, and $\mathcal{G}_K$ be a sentence's representation (by Definition 2), a question's representation (by Definition 3) and a representation of the corresponding required knowledge (by Definitions 4-9) respectively. Then*

(i) x is a constant node in $\mathcal{G}_S$ iff $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K) \models s\_const(x)$

(ii) x is a constant node in $\mathcal{G}_K$ iff $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K) \models k\_const(x)$

(iii) x is a constant node in $\mathcal{G}_Q$ iff $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K) \models q\_const(x)$

Proof 3 *(i) Let x be a constant node in $\mathcal{G}_S$. Therefore, by the definition of $s\_const(x)$ (Definition 10), there exists an outgoing edge labeled "$instance\_of$" from x to another node (say p) in $\mathcal{G}_S$. This means that $has\_s(x, instance\_of, p)$ is present in $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$ as a fact and hence $has\_s(x, instance\_of, p)$ is true in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$. Now, according to the rule* `n1` *in section 4.3.3 and part (a) of Proposition 4, $s\_const(x) \in \mathcal{M}$.*

Conversely, let $s\_const(x)$ is true in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$. Then, according to the part (b) of the Proposition 4, and the rule `n1` *in section 4.3.3, $\{has\_s(x, instance\_of, p)\} \subseteq \mathcal{M}$. In other words, x is an instance of p in $\mathcal{G}_S$. Therefore, according to the definition of $s\_const(x)$ (Definition 10), x is a constant node in $\mathcal{G}_S$.*

Hence, x is a constant node in $\mathcal{G}_S$ iff $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K) \models s\_const(x)$
Similarly, (ii) and (iii) are true with respect to the rules `n2` *and* `n3` *(Section 4.3.3) in $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$ respectively.*

Lemma 17 *Let $\mathcal{G}_S$, $\mathcal{G}_Q$, and $\mathcal{G}_K$ be a sentence's representation (by Definition 2), a question's representation (by Definition 3) and a representation of the corresponding required knowledge (by Definitions 4-9) respectively. Then a node y ($y \in \mathcal{G}_S$) is a cross-domain sibling of a node x ($x \in \mathcal{G}_K$) iff:*

$$\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K) \models k\_s\_crossdom\_sib(x, y)$$

Proof 4 *Let a node y in $\mathcal{G}_S$ is a cross-domain sibling of a node x in $\mathcal{G}_K$. Therefore, by the definition of $k\_s\_crossdom\_sib(x, y)$ (Definition 19), if y is an instance of i in $\mathcal{G}_S$ then x is an instance of i in $\mathcal{G}_K$, and both x and y are constant nodes in $\mathcal{G}_K$ and $\mathcal{G}_S$ respectively. This means that $has\_s(y, instance\_of, i)$, $has\_k(x, instance\_of, i)$, $s\_const(y)$ and $k\_const(x)$ are present in $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$ as facts and $not\_k\_s\_crossdom\_sib(x, y)$ is not present in $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$. Now, according to the rules `s1-s4` in section 4.3.3 and part (a) of the Proposition 4, $k\_s\_crossdom\_sib(x, y) \in \mathcal{M}$(the unique answer set of $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$).*

Conversely, let $k\_s\_crossdom\_sib(x, y) \in \mathcal{M}$, where $\mathcal{M}$ is the unique answer set of $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$. Then according to the part (b) of the Proposition 4, and rules `s1-s4` in section 4.3.3, $\{has\_s(y, instance\_of, i), has\_k(x, instance\_of, i), s\_const(y), k\_const(x)\} \subseteq \mathcal{M}$. Also, $not\_k\_s\_crossdom\_sib(x, y) \notin \mathcal{M}$. In other words, if y is an instance of i in $\mathcal{G}_S$ then x is an instance of i in $\mathcal{G}_K$, and both x and y are constant nodes in $\mathcal{G}_K$ and $\mathcal{G}_S$ respectively. Therefore, according to the definition of $k\_s\_crossdom\_sib(x, y)$ (Definition 19), the node y in $\mathcal{G}_S$ is a cross-domain sibling of the node x in $\mathcal{G}_K$.

Hence, a node y ($y \in \mathcal{G}_S$) is a cross-domain sibling of a node x ($x \in \mathcal{G}_K$) iff $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K) \models k\_s\_crossdom\_sib(x, y)$

Lemma 18 *Let $\mathcal{G}_S$, $\mathcal{G}_Q$, and $\mathcal{G}_K$ be a sentence's representation (by Definition 2), a question's representation (by Definition 3) and a representation of the corresponding required knowledge (by Definitions 4-9 respectively. Then*

(i) A node x in $\mathcal{G}_S$ has a parent iff $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K) \models s\_has\_par(x)$

(ii) A node x in $\mathcal{G}_K$ has a parent iff $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K) \models k\_has\_par(x)$

(iii) A node x in $\mathcal{G}_Q$ has a parent iff $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K) \models q\_has\_par(x)$

Proof 5 *(i) Let a node x in $\mathcal{G}_S$ has a parent. Therefore, by definition of $s\_has\_par(x)$ (Definition 13), there exists an incoming edge (say labeled r) from a constant node p to x in $\mathcal{G}_S$. This means that $has\_s(p, r, x)$, $s\_const(x)$ and $s\_const(p)$ are present in $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$ as facts and hence all of them are present in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$. Now, according to the rule `p1` in section 4.3.3 and part (a) of the Proposition 4, $s\_has\_par(x) \in \mathcal{M}$.*

Conversely, let $s\_has\_par(x) \in \mathcal{M}$, where $\mathcal{M}$ is the unique answer set of $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$. Then according to the part (b) of Proposition 4 and the rule `p1` in section 13, $\{has\_s(p, r, x), s\_const(x), s\_const(p)\} \subseteq \mathcal{M}$. In other words, x is a constant node in $\mathcal{G}_S$, p is a constant node in $\mathcal{G}_K$, and there exists an incoming edge from p to x in $\mathcal{G}_S$. Therefore, according to the definition of $s\_has\_par(x)$ (Definition 13), the node x in $\mathcal{G}_S$ has a parent.

Hence, a node x in $\mathcal{G}_S$ has a parent iff $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K) \models s\_has\_par(x)$.

Similarly, (ii) and (iii) above are true with respect to the rules `p2` and `p3` in $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$.

Lemma 19 *Let $\mathcal{G}_S$, $\mathcal{G}_Q$, and $\mathcal{G}_K$ be a sentence's representation (by Definition 2), a question's representation (by Definition 3) and a representation of the corresponding required knowledge (by Definitions 4-9) respectively. Then*

(i) A node x in $\mathcal{G}_S$ has a child iff $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K) \models s\_has\_child(x)$

(ii) A node x in $\mathcal{G}_K$ has a child iff $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K) \models k\_has\_child(x)$

(iii) A node x in $\mathcal{G}_Q$ has a child iff $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K) \models q\_has\_child(x)$

Proof 6 *(i) Let a node x in $\mathcal{G}_S$ has a child. Therefore, by definition of $s\_has\_child(x)$ (Definition 16), there exists an outgoing edge (say labeled r) from x to a constant node c in $\mathcal{G}_S$. This means that $has\_s(x,r,c)$, $s\_const(x)$ and $s\_const(c)$ are present in $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$ as facts and hence all of them are present in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$. Now, according to the rule `h1` in section 4.3.3 and part (a) of the Proposition 4, $s\_has\_child(x) \in \mathcal{M}$.*

Conversely, let $s\_has\_child(x) \in \mathcal{M}$, where $\mathcal{M}$ is the unique answer set of $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$. Then according to the part (b) of Proposition 4 and the rule `h1` in section 16, $\{has\_s(x,r,c), s\_const(x), s\_const(c)\} \subseteq \mathcal{M}$. In other words, x is a constant node in $\mathcal{G}_S$, c is a constant node in $\mathcal{G}_S$, and there exists an outgoing edge from x to c in $\mathcal{G}_S$. Therefore, according to the definition of $s\_has\_child(x)$ (Definition 16), the node x in $\mathcal{G}_S$ has a child.

Hence, a node x in $\mathcal{G}_S$ has a child iff $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K) \models s\_has\_child(x)$.

Similarly, (ii) and (iii) above are true with respect to the rules `h2` and `h3` in $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$.

Lemma 20 *Let $\mathcal{G}_S$, $\mathcal{G}_Q$, and $\mathcal{G}_K$ be a sentence's representation (by Definition 2), a question's representation (by Definition 3) and a representation of the corresponding required knowledge (by Definition 4-9) respectively. Then a node y in $\mathcal{G}_S$ is a cross-domain clone of a node x in $\mathcal{G}_K$ iff:*

$$\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K) \models k\_s\_crossdom\_clone(x,y)$$

Proof 7 *Let a node y in $\mathcal{G}_S$ is a cross-domain clone of a node x in $\mathcal{G}_K$. Therefore, according to the the definition of $k\_s\_crossdom\_clone(x,y)$ (Definition 20), exactly one of the following scenario is true.*

(a) y in $\mathcal{G}_S$ is a cross-domain sibling of x in $\mathcal{G}_K$, (p_j,r_j,x) is an edge in $\mathcal{G}_K$, (p'_j,r_j,y) is an edge in $\mathcal{G}_S$, p'_j is a cross-domain clone of p_j, (x,r_k,c_k) is an edge in $\mathcal{G}_K$, (y,r_k,c'_k) is an edge in $\mathcal{G}_S$, c'_k is a cross-domain sibling of c_k. This means that $k\_s\_crossdom\_sib(x,y)$, $has\_k(p_j,r_j,x)$, $has\_s(p'_j,r_j,y)$, $k\_s\_crossdom\_clone(p_j,p'_j)$, $has\_k(x,r_k,c_k)$, $has\_s(y,r_k,c'_k)$ and $k\_s\_crossdom\_sib(c_k,c'_k)$ are present in $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$ as facts and hence all of them are true in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$. Now, according to the rule `c1` in the section 4.3.3 and part (a) of the Proposition 4, $k\_s\_crossdom\_clone(x,y) \in \mathcal{M}$.

(b) y in $\mathcal{G}_S$ is a cross-domain sibling of x in $\mathcal{G}_K$, x does not have a parent, (x,r_k,c_k) is an edge in $\mathcal{G}_K$, (y,r_k,c'_k) is an edge in $\mathcal{G}_S$, c'_k is a cross-domain sibling of c_k. This means that $k\_s\_crossdom\_sib(x,y)$, $has\_k(x,r_k,c_k)$, $has\_s(y,r_k,c'_k)$ and $k\_s\_crossdom\_sib(c_k,c'_k)$ are present in $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$ as facts and hence all of them

are true in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$. Also, $k\_has\_par(x)$ is not true in $\mathcal{M}$. Now, according to the rule `c2` *in the section 4.3.3 and part (a) of the Proposition 4,*
$k\_s\_crossdom\_clone(x,y) \in \mathcal{M}$.

(c) *y in $\mathcal{G}_S$ is a cross-domain sibling of x in $\mathcal{G}_K$, (p_j,r_j,x) is an edge in $\mathcal{G}_K$, (p'_j,r_j,y) is an edge in $\mathcal{G}_S$, p'_j is a cross-domain clone of p_j and x does not have a child. This means that $k\_s\_crossdom\_sib(x,y)$, $has\_k(p_j,r_j,x)$, $has\_s(p'_j,r_j,y)$ and $k\_s\_crossdom\_clone(p_j,p'_j)$ are present as facts in $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$ and hence are true in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$. Also, $k\_has\_child(x)$ is not true in $\mathcal{M}$. Now, according to the rule* `c3` *in the section 4.3.3 and part (a) of the Proposition 4, $k\_s\_crossdom\_clone(x,y) \in \mathcal{M}$.*

(d) *y in $\mathcal{G}_S$ is a cross-domain sibling of x in $\mathcal{G}_K$, x does not have a parent and x does not have a child. This means that $k\_s\_crossdom\_sib(x,y)$ is present as a fact in $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$ and hence it is true in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$. Also, $k\_has\_par(x)$ and $k\_has\_child(x)$ are not true in $\mathcal{M}$. Now, according to the rule* `c4` *in the section 4.3.3 and part (a) of the Proposition 4, $k\_s\_crossdom\_clone(x,y) \in \mathcal{M}$.*

As shown above in all of the scenarios, $k\_s\_crossdom\_clone(x,y) \in \mathcal{M}$, i.e., $k\_s\_crossdom\_clone(x,y)$ belongs to the unique answer set of $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$.

Conversely, let $k\_s\_crossdom\_clone(x,y)$ is true in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$. Then according to the part (b) of the Proposition 4 and the rules `c1` *to* `c4` *in the section 4.3.3 any one of the following is true.*

(a) *$k\_s\_crossdom\_sib(x,y)$, $has\_k(p_j,r_j,x)$, $has\_s(p'_j,r_j,y)$, $k\_s\_crossdom\_clone(p_j,p'_j)$, $has\_k(x,r_k,c_k)$, $has\_s(y,r_k,c'_k)$ and $k\_s\_crossdom\_sib(c_k,c'_k)$ are true in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$. In other words, y in $\mathcal{G}_S$ is a cross-domain sibling of x in $\mathcal{G}_K$, (p_j,r_j,x) is an edge in $\mathcal{G}_K$, (p'_j,r_j,y) is an edge in $\mathcal{G}_S$, p'_j is a cross-domain clone of p_j, (x,r_k,c_k) is an edge in $\mathcal{G}_K$, (y,r_k,c'_k) is an edge in $\mathcal{G}_S$, and c'_k is a cross-domain sibling of c_k. Therefore according to the definition of $k\_s\_crossdom\_clone(x,y)$ (Definition 20), the node y in $\mathcal{G}_S$ is a cross-domain clone of the node x in $\mathcal{G}_K$.*

(b) *$k\_s\_crossdom\_sib(x,y)$, $has\_k(x,r_k,c_k)$, $has\_s(y,r_k,c'_k)$ and $k\_s\_crossdom\_sib(c_k,c'_k)$ are true in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$, and $k\_has\_par(x)$ is not true in $\mathcal{M}$. In other words, (x,r_k,c_k) is an edge in $\mathcal{G}_K$, (y,r_k,c'_k) is an edge in $\mathcal{G}_S$, c'_k is a cross-domain sibling of c_k and x does not have a parent. Therefore according to the definition of $k\_s\_crossdom\_clone(x,y)$ (Definition 20), the node y in $\mathcal{G}_S$ is a cross-domain clone of the node x in $\mathcal{G}_K$.*

(c) *$k\_s\_crossdom\_sib(x,y)$, $has\_k(p_j,r_j,x)$, $has\_s(p'_j,r_j,y)$ and $k\_s\_crossdom\_clone(p_j,p'_j)$ are true in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$ and $k\_has\_child(x)$ is not true in $\mathcal{M}$. In other words, y in $\mathcal{G}_S$ is a cross-domain sibling of x in $\mathcal{G}_K$, (p_j,r_j,x) is an edge in $\mathcal{G}_K$, (p'_j,r_j,y) is an edge in $\mathcal{G}_S$, p'_j is a cross-domain clone of p_j and x does not have a child. Therefore according to the* *definition of k_s_crossdom_clone(x,y) (Definition 20), the node y in $\mathcal{G}_S$ is a cross-domain clone of the node x in $\mathcal{G}_K$.*

(d) k_s_crossdom_sib(x,y) is true in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$, and k_has_par$(x)$ and k_has_child(x) are not true in $\mathcal{M}$. In other words, y in $\mathcal{G}_S$ is a cross-domain sibling of x in $\mathcal{G}_K$, x does not have a parent and x does not have a child. Therefore according to the definition of k_s_crossdom_clone(x,y) (Definition 20), the node y in $\mathcal{G}_S$ is a cross-domain clone of the node x in $\mathcal{G}_K$.

As shown above in all of the scenarios, the node y in $\mathcal{G}_S$ is a cross-domain clone of the node x in $\mathcal{G}_K$.

Hence, a node y in $\mathcal{G}_S$ is a cross-domain clone of a node x in $\mathcal{G}_K$ iff $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K) \models$ k_s_crossdom_clone(x,y).

Lemma 21 *Let $\mathcal{G}_S$, $\mathcal{G}_Q$, and $\mathcal{G}_K$ be a sentence's representation (by Definition 2), a question's representation (by Definition 3) and a representation of the corresponding required knowledge (by Definition 4-9) respectively. Then, there exists a merged representation $\mathcal{G}_M$ such that there exists an edge $(h,l,t) \in \mathcal{G}_M$ iff:*

$$\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K) \models has\_m(h,l,t)$$

Proof 8 *Let (h,l,t) is an edge in $\mathcal{G}_M$. Therefore, by definition of $\mathcal{G}_M$ (Definition 21), exactly one of the following scenarios is true.*

(a) (h,l,t) is an edge in $\mathcal{G}_S$. This means that has_s(h,l,t) is present in $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$ as a fact and hence has_s(h,l,t) is true in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$. Now, according to the rule `m1` in the section 4.3.3 and part (a) of the Proposition 4, has_m$(h,l,t) \in \mathcal{M}$

(b) $(h,l,t1)$ is an edge in $\mathcal{G}_S$, $(h2,l2,t)$ is an edge in $\mathcal{G}_S$, $t \neq t1$, $t1$ is a cross-domain clone of t_k from $\mathcal{G}_K$ to $\mathcal{G}_S$ and t is a cross-domain clone of a node t_k from $\mathcal{G}_K$ to $\mathcal{G}_S$. This means that has_s$(h,l,t1)$, has_$(h2,l2,t)$, k_s_crossdom_clone$(t_k,t1)$ and k_s_crossdom_clone(t_k,t) are present as facts in $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$ and all the constant nodes in $\mathcal{G}_K$ have cross-domain clones in $\mathcal{G}_S$. Hence all of the above are true in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$. Now, according to the rules `m2-m5` in the section 4.3.3 and part (a) of the Proposition 4, has_m$(h,l,t) \in \mathcal{M}$

(c) $(h,l2,t2)$ is an edge in $\mathcal{G}_S$, $(h1,l,t)$ is an edge in $\mathcal{G}_S$, $h \neq h1$, $h1$ is a cross-domain clone of a node h_k from $\mathcal{G}_K$ to $\mathcal{G}_S$ and h is a cross-domain clone of h_k from $\mathcal{G}_K$ to $\mathcal{G}_S$. This means that has_s$(h,l2,t2)$, has_$(h1,l,t)$, k_s_crossdom_clone$(h_k,h1)$ and k_s_crossdom_clone(h_k,h) are present as facts in $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$ and all the constant nodes in $\mathcal{G}_K$ have cross-domain clones in $\mathcal{G}_S$. Hence all of the above are true in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$. Now, according to the rules `m2-m4` and `m6` in the section 4.3.3 and part (a) of the Proposition 4, has_m$(h,l,t) \in \mathcal{M}$

As shown in all the scenarios above, has_m$(h,l,t) \in \mathcal{M}$, i.e., if (h,l,t) is an edge in the merged representation $\mathcal{G}_M$ then has_m(h,l,t) belongs to the unique answer set of $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$. Conversely, let has_m$(h,l,t)$ is true in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$. Then according to the part (b) of the Proposition 4 and the rules `m1` to `m6` in the section 4.3.3 any one of the following scenarios must be true.

(a) *$has\_s(h,l,t)$ is true in $\mathcal{M}$. In other words, (h,l,t) is an edge in $\mathcal{G}_S$. Therefore, according to the definition of $\mathcal{G}_M$ (Definition 21), (h,l,t) is an edge in $\mathcal{G}_M$.*

(b) *$has\_s(h,l,t1) \in \mathcal{M}$, $has\_(h2,l2,t) \in \mathcal{M}$, $k\_s\_crossdom\_clone(t_k,t1) \in \mathcal{M}$, $k\_s\_crossdom\_clone(t_k,t) \in \mathcal{M}$, $t \neq t1$ and $k\_all\_covered \in \mathcal{M}$. In other words, $(h,l,t1)$ is an edge in $\mathcal{G}_S$, $(h2,l2,t)$ is an edge in $\mathcal{G}_S$, $t1$ is a cross-domain clone of t_k from $\mathcal{G}_K$ to $\mathcal{G}_S$, t is a cross-domain clone of t_k from $\mathcal{G}_K$ to $\mathcal{G}_S$, $t \neq t1$ and all the constant nodes in $\mathcal{G}_K$ have cross-domain clones in $\mathcal{G}_S$. Therefore, according to the definition of $\mathcal{G}_M$ (Definition 21), (h,l,t) is an edge in $\mathcal{G}_M$.*

(c) *$has\_s(h,l2,t2) \in \mathcal{M}$, $has\_(h1,l,t) \in \mathcal{M}$, $k\_s\_crossdom\_clone(h_k,h1) \in \mathcal{M}$, $k\_s\_crossdom\_clone(h_k,h) \in \mathcal{M}$, $h \neq h1$ and $k\_all\_covered \in \mathcal{M}$. In other words, $(h,l2,t2)$ is an edge in $\mathcal{G}_S$, $(h1,l,t)$ is an edge in $\mathcal{G}_S$, $h1$ is a cross-domain clone of h_k from $\mathcal{G}_K$ to $\mathcal{G}_S$, h is a cross-domain clone of h_k from $\mathcal{G}_K$ to $\mathcal{G}_S$, $h \neq h1$ and all the constant nodes in $\mathcal{G}_K$ have cross-domain clones in $\mathcal{G}_S$. Therefore, according to the definition of $\mathcal{G}_M$ (Definition 21), (h,l,t) is an edge in $\mathcal{G}_M$.*

As shown above, in all the scenarios, if $has\_m(h,l,t) \in \mathcal{M}$ then (h,l,t) is an edge in the merged representation $\mathcal{G}_M$.

Hence, an edge (h,l,t) is an edge in the merged representation $\mathcal{G}_M$ iff $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K) \models has\_m(h,l,t)$

Lemma 22 *Let $\mathcal{G}_S$, $\mathcal{G}_Q$, and $\mathcal{G}_K$ be a sentence's representation (by Definition 2), a question's representation (by Definition 3) and a representation of the corresponding required knowledge (by Definitions 4-9) respectively. Also, $\mathcal{G}_M$ is the merged representation of $\mathcal{G}_S$ and $\mathcal{G}_K$ (By definition 10). Then, a node x is a constant node in $\mathcal{G}_M$ iff:*

$$\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K) \models m\_const(x)$$

Proof 9 *Let x be a constant node in $\mathcal{G}_M$. Therefore, by the definition of $m\_const(x)$ (Definition 22), there exists an outgoing edge labeled "$instance\_of$" from x to another node (say p) in $\mathcal{G}_M$. This means that $has\_m(x,instance\_of,p)$ is present in $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$ as a fact and hence according to the part (a) Proposition 4, $has\_m(x,instance\_of,p)$ is present in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$. Now, according to the rule `n3` in section 4.3.3 and part (a) of Proposition 4, $m\_const(x) \in \mathcal{M}$.*

Conversely, let $m\_const(x)$ is present in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$. Then, according to the part (b) of the Proposition 4, and the rule `n3` in section 4.3.3, $\{has\_m(x,instance\_of,p)\} \subseteq \mathcal{M}$. In other words, x is an instance of p in $\mathcal{G}_M$. Therefore, according to the definition of $m\_const(x)$ (Definition 22), x is a constant node in $\mathcal{G}_M$.

Hence, x is a constant node in $\mathcal{G}_M$ iff $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K) \models m\_const(x)$

Lemma 23 *Let $\mathcal{G}_S$, $\mathcal{G}_Q$, and $\mathcal{G}_K$ be a sentence's representation (by Definition 2), a question's representation (by Definition 3) and a representation of the corresponding required knowledge (by Definitions 4-9) respectively. Also, $\mathcal{G}_M$ is the merged representation of $\mathcal{G}_S$ and $\mathcal{G}_K$ (By Definition 21). Then, a node y ($y \in \mathcal{G}_M$) is a cross-domain sibling of a node x ($x \in \mathcal{G}_Q$) iff:*

$$\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K) \models q\_m\_crossdom\_sib(x,y)$$

Proof 10 *Let a node y in $\mathcal{G}_M$ is a cross-domain sibling of a node x in $\mathcal{G}_Q$. Therefore, by the definition of $q\_m\_crossdom\_sib(x,y)$ (Definition 25), if y is an instance of i in $\mathcal{G}_M$ then x is an instance of i in $\mathcal{G}_Q$, and both x and y are constant nodes in $\mathcal{G}_Q$ and $\mathcal{G}_M$ respectively. This means that $has\_m(y, instance\_of, i)$, $has\_q(x, instance\_of, i)$, $m\_const(y)$ and $q\_const(x)$ are present in $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$ as facts and $not\_q\_m\_crossdom\_sib(x,y)$ is not in $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$. Now, according to the rules `s5-s8` in section 4.3.3 and part (a) of the Proposition 4, $q\_m\_crossdom\_sib(x,y) \in \mathcal{M}$ (the unique answer set of $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$).*

Conversely, let $q\_m\_crossdom\_sib(x,y) \in \mathcal{M}$, where $\mathcal{M}$ is the unique answer set of $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$. Then according to the part (b) of the Proposition 4, and the rules `s5-s8` in the section 4.3.3, $\{has\_m(y, instance\_of, i), has\_q(x, instance\_of, i), m\_const(y), q\_const(x)\} \subseteq \mathcal{M}$. Also, $not\_q\_m\_crossdom\_sib(x,y) \notin \mathcal{G}_M$. In other words, if y is an instance of i in $\mathcal{G}_M$ then x is an instance of i in $\mathcal{G}_Q$, and both x and y are constant nodes in $\mathcal{G}_Q$ and $\mathcal{G}_M$ respectively. Therefore, according to the definition of $q\_m\_crossdom\_sib(x,y)$ (Definition 25), the node y in $\mathcal{G}_M$ is a cross-domain sibling of the node x in $\mathcal{G}_Q$.

Hence, a node y ($y \in \mathcal{G}_M$) is a cross-domain sibling of a node x ($x \in \mathcal{G}_Q$) iff $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K) \models q\_m\_crossdom\_sib(x,y)$

Lemma 24 *Let $\mathcal{G}_S$, $\mathcal{G}_Q$, and $\mathcal{G}_K$ be a sentence's representation (by Definition 2), a question's representation (by Definition 3) and a representation of the corresponding required knowledge (by Definitions 4-9) respectively. Also, $\mathcal{G}_M$ is the merged representation of $\mathcal{G}_S$ and $\mathcal{G}_K$ (By Definition 21). Then, a node x in $\mathcal{G}_M$ has a parent iff:*

$$\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K) \models m\_has\_par(x)$$

Proof 11 *Let a node x in $\mathcal{G}_M$ has a parent. Therefore, by definition of $m\_has\_par(x)$ (Definition 23), there exists an incoming edge (say labeled r) from a constant node p to x in $\mathcal{G}_M$. This means that $has\_m(p,r,x)$, $m\_const(x)$ and $m\_const(p)$ are present in $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$ as facts and hence all of them are present in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$. Now, according to the rule `p4` in section 4.3.3 and part (a) of the Proposition 4, $m\_has\_par(x) \in \mathcal{M}$.*

Conversely, let $m\_has\_par(x) \in \mathcal{M}$, where $\mathcal{M}$ is the unique answer set of $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$. Then according to the part (b) of Proposition 4 and the rule `p4` in section 4.3.3, $\{has\_m(p,r,x), m\_const(x), m\_const(p)\} \subseteq \mathcal{M}$. In other words, x is a constant node in $\mathcal{G}_M$, p is a constant node in $\mathcal{G}_M$, and there exists an incoming edge from p to x in $\mathcal{G}_M$. Therefore, according to the definition of $m\_has\_par(x)$ (Definition 13), the node x in $\mathcal{G}_M$ has a parent.

Hence, a node x in $\mathcal{G}_M$ has a parent iff $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K) \models m\_has\_par(x)$.

Lemma 25 *Let $\mathcal{G}_S$, $\mathcal{G}_Q$, and $\mathcal{G}_K$ be a sentence's representation (by Definition 2), a question's representation (by Definition 3) and a representation of the corresponding required knowledge (by Definitions 4-9) respectively. Also, $\mathcal{G}_M$ is the merged representation of $\mathcal{G}_S$ and $\mathcal{G}_K$ (By Definition 21). Then, a node x in $\mathcal{G}_M$ has a child iff:*

$$\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K) \models m\_has\_child(x)$$

Proof 12 *Let a node x in $\mathcal{G}_M$ has a child. Therefore, by definition of $m\_has\_child(x)$ (Definition 24), there exists an outgoing edge (say labeled r) from x to a constant node c in $\mathcal{G}_M$. This means that $has\_m(x,r,c)$, $m\_const(x)$ and $m\_const(c)$ are present in $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$ as facts and hence all of them are present in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$. Now,* *according to the rule* h4 *in section 4.3.3 and part (a) of the Proposition 4,* $m\_has\_child(x) \in \mathcal{M}$.

Conversely, let $m\_has\_child(x) \in \mathcal{M}$*, where* $\mathcal{M}$ *is the unique answer set of* $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$*. Then according to the part (b) of Proposition 4 and the rule* h4 *in section 4.3.3,* $\{has\_m(x,r,c), m\_const(x), m\_const(c)\} \subseteq \mathcal{M}$*. In other words, x is a constant node in* $\mathcal{G}_M$*, c is a constant node in* $\mathcal{G}_M$*, and there exists an outgoing edge from x to c in* $\mathcal{G}_M$*. Therefore, according to the definition of* $m\_has\_child(x)$ *(Definition 24), the node x in* $\mathcal{G}_M$ *has a child.*

Hence, a node x in $\mathcal{G}_M$ *has a child iff* $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K) \vDash m\_has\_child(x)$.

Lemma 26 *Let* $\mathcal{G}_S$*,* $\mathcal{G}_Q$*, and* $\mathcal{G}_K$ *be a sentence's representation (by Definition 2), a question's representation (by Definition 3) and a representation of the corresponding required knowledge (by Definitions 4-9) respectively. Also,* $\mathcal{G}_M$ *is the merged representation of* $\mathcal{G}_S$ *and* $\mathcal{G}_K$ *(By Definition 21). Then, a node x in* $\mathcal{G}_Q$ *has a cross-domain clone y in* $\mathcal{G}_M$ *iff:*

$$\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K) \vDash q\_m\_crossdom\_clone(x,y)$$

Proof 13 *Let a node y in* $\mathcal{G}_M$ *is a cross-domain clone of a node x in* $\mathcal{G}_Q$*. Therefore, according to the the definition of* $q\_m\_crossdom\_clone(x,y)$ *(Definition 26), exactly one of the following scenario is true.*

(a) *y in* $\mathcal{G}_M$ *is a cross-domain sibling of x in* $\mathcal{G}_Q$*,* (p_j, r_j, x) *is an edge in* $\mathcal{G}_Q$*,* (p'_j, r_j, y) *is an edge in* $\mathcal{G}_M$*,* p'_j *is a cross-domain clone of* p_j*,* (x, r_k, c_k) *is an edge in* $\mathcal{G}_Q$*,* (y, r_k, c'_k) *is an edge in* $\mathcal{G}_M$*,* c'_k *is a cross-domain sibling of* c_k*. This means that* $q\_m\_crossdom\_sib(x,y)$*,* $has\_k(p_j, r_j, x)$*,* $has\_m(p'_j, r_j, y)$*,* $q\_m\_crossdom\_clone(p_j, p'_j)$*,* $has\_q(x, r_k, c_k)$*,* $has\_m(y, r_k, c'_k)$ *and* $q\_m\_crossdom\_sib(c_k, c'_k)$ *are present in* $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$ *as facts and hence all of them are true in the unique answer set* $\mathcal{M}$ *of* $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$*. Now, according to the rule* c5 *in the section 4.3.3 and part (a) of the Proposition 4,* $q\_m\_crossdom\_clone(x,y) \in \mathcal{M}$.

(b) *y in* $\mathcal{G}_M$ *is a cross-domain sibling of x in* $\mathcal{G}_Q$*, x does not have a parent,* (x, r_k, c_k) *is an edge in* $\mathcal{G}_Q$*,* (y, r_k, c'_k) *is an edge in* $\mathcal{G}_M$*,* c'_k *is a cross-domain sibling of* c_k*. This means that* $q\_m\_crossdom\_sib(x,y)$*,* $has\_q(x, r_k, c_k)$*,* $has\_m(y, r_k, c'_k)$ *and* $q\_m\_crossdom\_sib(c_k, c'_k)$ *are present in* $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$ *as facts and hence all of them are true in the unique answer set* $\mathcal{M}$ *of* $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$*. Also,* $q\_has\_par(x)$ *is not true in* $\mathcal{M}$*. Now, according to the rule* c6 *in the section 4.3.3 and part (a) of the Proposition 4,* $q\_m\_crossdom\_clone(x,y) \in \mathcal{M}$.

(c) *y in* $\mathcal{G}_M$ *is a cross-domain sibling of x in* $\mathcal{G}_Q$*,* (p_j, r_j, x) *is an edge in* $\mathcal{G}_Q$*,* (p'_j, r_j, y) *is an edge in* $\mathcal{G}_M$*,* p'_j *is a cross-domain clone of* p_j *and x does not have a child. This means that* $q\_m\_crossdom\_sib(x,y)$*,* $has\_q(p_j, r_j, x)$*,* $has\_m(p'_j, r_j, y)$ *and* $q\_m\_crossdom\_clone(p_j, p'_j)$ *are present as facts in* $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$ *and hence are true in the unique answer set* $\mathcal{M}$ *of* $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$*. Also,* $q\_has\_child(x)$ *is not true in* $\mathcal{M}$*. Now, according to the rule* c7 *in the section 4.3.3 and part (a) of the Proposition 4,* $q\_m\_crossdom\_clone(x,y) \in \mathcal{M}$.

(d) *y in $\mathcal{G}_M$ is a cross-domain sibling of x in $\mathcal{G}_Q$, x does not have a parent and x does not have a child. This means that $q\_m\_crossdom\_sib(x,y)$ is present as a fact in $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$ and hence it is true in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$. Also, $q\_has\_par(x)$ and $q\_has\_child(x)$ are not true in $\mathcal{M}$. Now, according to the rule* `c8` *in the section 4.3.3 and part (a) of the Proposition 4, $q\_m\_crossdom\_clone(x,y) \in \mathcal{M}$.*

As shown above in all of the scenarios, $q\_m\_crossdom\_clone(x,y) \in \mathcal{M}$, i.e., $q\_m\_crossdom\_clone(x,y)$ belongs to the unique answer set of $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$. Conversely, let $q\_m\_crossdom\_clone(x,y)$ is true in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$. Then according to the part (b) of the Proposition 4 and the rules `c1` *to* `c4` *in the section 4.3.3 any one of the following is true.*

(a) *$q\_m\_crossdom\_sib(x,y)$, $has\_q(p_j,r_j,x)$, $has\_m(p'_j,r_j,y)$, $q\_m\_crossdom\_clone(p_j,p'_j)$, $has\_q(x,r_k,c_k)$, $has\_m(y,r_k,c'_k)$ and $q\_m_crossdom\_sib(c_k,c'_k)$ are true in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$. In other words, y in $\mathcal{G}_M$ is a cross-domain sibling of x in $\mathcal{G}_Q$, (p_j,r_j,x) is an edge in $\mathcal{G}_Q$, (p'_j,r_j,y) is an edge in $\mathcal{G}_M$, p'_j is a cross-domain clone of p_j, (x,r_k,c_k) is an edge in $\mathcal{G}_Q$, (y,r_k,c'_k) is an edge in $\mathcal{G}_M$, and c'_k is a cross-domain sibling of c_k. Therefore according to the definition of $q\_m\_crossdom\_clone(x,y)$ (Definition 26), the node y in $\mathcal{G}_M$ is a cross-domain clone of the node x in $\mathcal{G}_Q$.*

(b) *$q\_m\_crossdom\_sib(x,y)$, $has\_q(x,r_k,c_k)$, $has\_m(y,r_k,c'_k)$ and $q\_m\_crossdom\_sib(c_k,c'_k)$ are true in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$, and $q\_has\_par(x)$ is not true in $\mathcal{M}$. In other words, (x,r_k,c_k) is an edge in $\mathcal{G}_Q$, (y,r_k,c'_k) is an edge in $\mathcal{G}_M$, c'_k is a cross-domain sibling of c_k and x does not have a parent. Therefore according to the definition of $q\_m\_crossdom\_clone(x,y)$ (Definition 26), the node y in $\mathcal{G}_M$ is a cross-domain clone of the node x in $\mathcal{G}_Q$.*

(c) *$q\_m\_crossdom\_sib(x,y)$, $has\_q(p_j,r_j,x)$, $has\_m(p'_j,r_j,y)$ and $q\_m\_crossdom\_clone(p_j,p'_j)$ are true in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$ and $q\_has\_child(x)$ is not true in $\mathcal{M}$. In other words, y in $\mathcal{G}_M$ is a cross-domain sibling of x in $\mathcal{G}_Q$, (p_j,r_j,x) is an edge in $\mathcal{G}_Q$, (p'_j,r_j,y) is an edge in $\mathcal{G}_M$, p'_j is a cross-domain clone of p_j and x does not have a child. Therefore according to the definition of $q\_m\_crossdom\_clone(x,y)$ (Definition 26), the node y in $\mathcal{G}_M$ is a cross-domain clone of the node x in $\mathcal{G}_Q$.*

(d) *$q\_m\_crossdom\_sib(x,y)$ is true in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K)$, and $q\_has\_par(x)$ and $q\_has\_child(x)$ are not true in $\mathcal{M}$. In other words, y in $\mathcal{G}_M$ is a cross-domain sibling of x in $\mathcal{G}_Q$, x does not have a parent and x does not have a child. Therefore according to the definition of $q\_m\_crossdom\_clone(x,y)$ (Definition 26), the node y in $\mathcal{G}_M$ is a cross-domain clone of the node x in $\mathcal{G}_Q$.*

As shown above in all of the scenarios, the node y in $\mathcal{G}_M$ is a cross-domain clone of the node x in $\mathcal{G}_Q$.

Hence, a node y in $\mathcal{G}_M$ is a cross-domain clone of a node x in $\mathcal{G}_Q$ iff $\Pi(\mathcal{G}_S,\mathcal{G}_Q,\mathcal{G}_K) \models q\_m\_crossdom\_clone(x,y)$.

Theorem 2 *Let $\mathcal{G}_S$, $\mathcal{G}_Q$, and $\mathcal{G}_K$ be a sentence's representation (by Definition 2), a question's representation (by Definition 3) and a representation of the corresponding required knowledge (by Definitions 4-9) respectively. Also, $\mathcal{G}_M$ is the merged representation of $\mathcal{G}_S$ and $\mathcal{G}_K$ (Obtained by Definition 21). Then, an unknown node q_i in $\mathcal{G}_Q$ has an answer x ($x \in \mathcal{G}_M$) iff:*

$$\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K) \vDash ans(q\_i, x)$$

Proof 14 *Let q_i be an unknown node in $\mathcal{G}_Q$ and x be its answer in $\mathcal{G}_M$. Therefore, according to the definition of $ans(q_i, x)$ (Definition 27), x is a cross-domain clone of q_i and the edge $(q\_i, instance\_of, q) \in \mathcal{G}_Q$. This means that $crossdom\_clone(q, m, q\_i, x)$, and $has\_q(q\_i, instance\_of, q)$ are present as facts in $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$, and hence they are true in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$. Now, according to the rule `a1` in section 4.3.4 and part (a) of the Proposition 4, $ans(q_i, x) \in \mathcal{M}$.*

Conversely, let $ans(q\_i, x)$ is true in the unique answer set $\mathcal{M}$ of $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K)$. Then, according to the part (b) of the Proposition 4 and the rule `a1` in the section 4.3.4, $\{crossdom\_clone(q, m, q\_i, x), has\_q(q\_i, instance\_of, q)\} \subseteq \mathcal{M}$. In other words, x is a cross-domain clone of q_i from $\mathcal{G}_Q$ to $\mathcal{G}_M$ and the edge $(q\_i, instance\_of, q) \in \mathcal{G}_Q$. Therefore, according to the definition of $ans(q_i, x)$ (Definition 27), the node x in $\mathcal{G}_M$ is an answer to the unknown node q_i.

Hence an unknown node q_i in $\mathcal{G}_Q$ has an answer x ($x \in \mathcal{G}_M$) iff $\Pi(\mathcal{G}_S, \mathcal{G}_Q, \mathcal{G}_K) \vDash ans(q\_i, x)$.