2. Commonsense Knowledge Categorization

In this section, we present the new commonsense knowledge categories discovered as part of this work. We used the Winograd Schema Challenge sentences and questions to identify these categories.

2.1 An Example Knowledge

Let us consider a Winograd Schema Challenge problem as shown below.

Sentence: The man could not lift his son because he was so weak.
Question: What was so weak?
Answer: The man

To correctly solve the above mentioned problem, the following commonsense knowledge is required.
Required Knowledge:
A person X could not lift something Y may be because X was weak.

In other words, the required knowledge is,
IF X (a person) was weak **prevents** Y (a person) to lift **THEN** X=Y

In this work, we represent the above knowledge as,

```
P [instance_of: weak,
   is_trait_of: X [instance_of: person]]
prevents
A [instance_of: lift,
   agent: Y [instance_of: person]]
implies
X=Y
```

Here, `P [instance_of: weak, is_trait_of: X [instance_of: person]]` represents a property, `X [instance_of: person]` and `Y [instance_of: person]` represent entities and `A [instance_of: lift, agent: Y [instance_of: person]]` represents an event. A property, an event and an entity are the building blocks of the commonsense knowledge mentioned in this work. The detailed description of each of them is provided in the section below.

2.2 Background

In this section, we explain the basic concepts and important representation aspects of each knowledge type identified in this work. There are four basic concepts and two representation aspects. The basic concepts are as defined below.

1. An Entity: All commonsense categories defined in this work are centered around the causal and preventive relationships among actions (verbs) and properties (adjectives) in the English language. An entity is a concrete or abstract thing that participates in an action or/and has a property associated with it. For example, in the knowledge shown in the example above, `X` and `Y` are entities as they either participate in an action or have some property associated with them. Neither an action nor a property can be categorized as an entity. Following are the other important attributes of an entity in this work.

1. For every entity X_i, there exists at least one conceptual class C_i such that X_i is of the type C_i. This information is represented with the help of an *instance_of* relationship from X_i and C_i. Every entity belongs to the *entity* class by default. For example, let X be an entity such that X is a *person*, then X is represented in Neo-Davidsonian fashion as

   ```
   X [instance_of: person, instance_of: entity]
   ```

2. An entity may have a list of properties associated with it via the relation *trait*. These properties collectively alter the overall meaning of the entity. For example, let X be an entity, X is a person, and P_1, P_2 be the properties associated with it, then X is represented in Neo-Davidsonian fashion as,

   ```
   X [instance_of: person, instance_of: entity,
           trait: P1, trait: P2]
   ```

3. An entity may be related to other entities via a predefined set of relations. For this work, these relations are the relations between two entities as defined in an off the shelf semantic parsing system called K-Parser[1]. Let X_1 and X_2 be two entities and there exists a relation from X_1 to X_2 with the label *related_to*, then X_1 is represented in Neo-Davidsonian fashion as below.

   ```
   X1 [instance_of: entity, related_to: X2
                                       [instance_of: entity]]
   ```

Based on the above description, the general Neo-Davidsonian style representation of an entity is as follows.

```
X [instance_of: C1,..., instance_of: Ci,
   trait: P1,..., trait: Pn,
   rel1: X1,..., relm: Xm]
```

Where, `X` is an entity of types C_1 to C_i, has traits P_1 to P_n and related to the entities X_1 to X_m via directed relations labeled rel_1 to rel_m

2. A Property: A property is something that alters the meaning of an entity by either adding a special positive or negative quality to the entity. In Neo-Davidsonian fashion, a variable is used to represent an entity. For example, let X be an entity, and *weak* be the property associated with X, then a variable (say P) is used to represent the property *weak*. A property is also represented in Neo-Davidsonian fashion is as below.

```
P [instance_of: weak, is_trait_of: X]
```

Here, *instance_of* is the relation between the variable P, representing a property, and the conceptual class of P i.e., the actual value of the property. For example, *weak*. *is_trait_of* is the relation between the variable P and the entity X.

1 www.kparser.org

Following are the other important attributes of a property in this work.

1. A property belongs to only one conceptual class which is also its actual value.
2. A property can not exists by itself. It always occurs with an entity.
3. A property may be associated with more than one entity at a time.

The general Neo-Davidsonian style representation of a property is as follows.

```
P [instance_of: PROP,
   is_trait_of: X1,
   ...,
   is_trait_of: Xn]
```

Where, `P` is a property with conceptual class `PROP`, and it is associated with the entities X_1 to X_n through multiple copies of the `is_trait_of` relation.

3. An Event: An event is a complex representation of an action and a set of entities that participate in that action. Similar to the representation of a property, an event is also represented by a variable (Neo-Davidsonian style). Following are the important attributes of an event.

1. Every event has a driver action. The driver action may be a uni-gram or a phrasal verb. Just like an entity and a property, an even also has a conceptual class. The only conceptual class of an event is defined by its driver action. For example, let "*X (a an entity) lifts Y (an entity)*" be an event such that it is represented by the variable E and *lift* be the driver action of E, then the driver action aspect of E is represented as below.

   ```
   E [instance_of: lift]
   ```

2. Every event has zero or more entities participating in it in different capacities or roles. For example, let "*X (an entity) lift Y (an entity)*" be an event such that it is represented by the variable E, *lift* be the driver action of E, *X* and *Y* are entities that are participating in E and playing the roles of *agent* and *recipient* in E respectively. The roles in this example are borrowed from the K-Parser1 definition. Then, E is represented as below.

   ```
   E [instance_of: lift,
      agent: X [instance_of: entity],
      recipient: Y [instance_of: entity]]
   ```

   Where, `X [instance_of: entity]` and `Y [instance_of: entity]` are the representations of the entities *X* and *Y* respectively. The roles or relationship labels (`agent` and `recipient` in the above example) that each entity plays in an event are taken from the set of relations between an event node and an entity node in the K-Parser[1].

Based on the above description the general Neo-Davidsonian style representation of an event is as shown below.

```
E [instance_of: A,
   rel1: X1,
   ...,
   reln: Xn]
```

Where, `E` is an event with driver action `A`, and the entities X_1 to X_n play the roles rel_1 to rel_n in `A`. rel_1 to rel_n are taken from the set of relations between an event node and an entity node in the K-Parser[1].

4. A Conceptual Class: A conceptual class defines the type of another basic concept (an entity or a property or an event). Following are the attributes of a conceptual class with respect to the other basic concepts.

- The conceptual class of a basic concept is connected to it via directed instance of relation.
- There exists at least one conceptual class for each entity. By default all the entities belong to the entity class.
- There exists exactly one conceptual class of a property i.e., the base form of the adjective that defines the property.
- There exists exactly one conceptual class of an event i.e., its driver action.

Following are the important representation aspects of knowledge types defined in this work.
(A) Causal and Sequential Relations: An important representation aspect of the commonsense knowledge identified in this work is a set of relations between events and properties. Following are the relations and their example.

- **Causality based relations:** There are two causality based directed relations, *causes* and *prevents*. As the name suggests,
  X causes Y represents that *X* causes *Y*,
  X prevents Y represents that *X* prevents *Y* from existing,
- **Sequential relation:** There is one sequential relation, *followed by*. As the name suggests,
  X followed by Y represents that *Y* occurs after *X*.

More details on the semantics of these relations are mentioned in the sections below while defining the different commonsense knowledge categories.

(B) Commonsense Knowledge Format: The general formal for each of the commonsense knowledge categories defined in this work is as follows.

```
S implies X=Y
```

Here, `S` is a set of events and properties connected via a chain of sequential/causal relations and both `X` and `Y` are entities that participate either in two different events in `S` or in an event and a property in `S`.

The keyword **`implies`** is equivalent to a logical implication which state that if all the conditions in the left hand side of the implication are satisfied then the right hand side must be satisfied. In terms of the commonsense knowledge, if the chain of events and properties in `S` are true then `X=Y`.

2.3 Commonsense Knowledge Types

We identified 12 different knowledge types in this work. 10 out of 12 knowledge types are based on causality and sequential relations between a property and an event. The remaining two are based on specific events, entities and probabilistic comparison between the occurrences of two propositions.

1. Knowledge Type 1: A Property Prevents an Event
In this category, if a property P prevents an event E from executing then an entity associated with P is also a participant in E. Formally, the generalized representation of this category is as follows.

```
P [instance_of: PROP, is_trait_of: X1,..., is_trait_of: Xm]
prevents
E [instance_of: A, rel1: Y1,..., reln: Yn]
implies
X=Y
```

where, m, n≥ 1, X $\in \bigcup_{i=1}^{m} \{X_i\}$, and Y $\in \bigcup_{j=1}^{n} \{Y_j\}$

Intuitively, a statement of the above category means that, if a property P prevents an event E then an entity X associated with P is same as the entity Y that participates in E.
A winograd schema example that requires the knowledge from above category is mentioned below.

Example 1

Sentence: *The man couldn't lift his son because he was so weak.*
Question: *Who was weak?*
Answers: *The man.*
Required Knowledge:

```
P [instance-of: weak,
   is_trait_of: X [instance-of:person]]
prevents
E [instance_of: lift;
   agent: Y1 [instance-of:person]]
implies
X=Y1
```

2. Knowledge Type 2: A Sequence of Events (E_{seq}) Causes an Event (E_{n+1}) In this category of commonsense knowledge if a sequence of events E_{seq} causes another event E_{n+1} then an entity X that participates in an event in E_{seq} also participates in the event E_{n+1}. Formally, the generalized representation of this category is as follows.

```
[+/-] E_1 [instance_of: A_1,
          rel_11: X_11, ..., rel_1i_1: X_1i_1]
followed by
...
[+/-] E_n [instance_of: A_n,
          rel_n1: X_n1, ..., rel_ni_n: X_ni_n]
causes
[+] E_(n+1) [instance_of: A_(n+1),
          rel_(n+1)1: X_(n+1)1, ..., rel_(n+1)j: X_(n+1)j]
implies
X = Y
```

where, $n \geq 1$, X$\in \bigcup_{a=1}^{n} \bigcup_{k=1}^{i_a} \{X_{ak}\}$, and Y$\in \bigcup_{l=1}^{j} \{X_{(n+1)l}\}$

The symbol `[+/-]` in front of an event E means that E may execute or it is in-executable (not executed). In other words, the instances of the knowledge category shown by the representation above include all the combinations of executable and in-executable versions of the events E_1 to E_n. Similarly `[+]` in front of an event E means that E is executed and `[-]` in front of an event E means that E is in-executable (or not executed). To make the representation simpler, if an event is executable we may not add a `[+]` symbol in front of it i.e. writing "`[+] E`" is same as writing "`E`".

Intuitively, a knowledge of the above category means that, If a sequence of events (executable or in-executable) (E_{seq} = E_1,..., E_n) causes the execution of another event (E_{n+1}), then an entity X participating in an event in E_{seq} is same as an entity Y participating in E_{n+1}.

A winograd schema example that requires a knowledge instance belonging to the above mentioned category is as shown below.

Example 1:

Sentence: *The city councilmen refused the demonstrators a permit because they feared violence.*
Question: *Who feared violence?*
Answers: *The city councilmen*
Required Knowledge:

```
[+] E_1 [instance_of: fear;
        agent: X [instance-of:entity];
        recipient: A [instance_of: violence]]
causes
[+] E_2 [instance_of: refused;
        agent: Y [instance-of:entity];
        object: B [instance_of: permit]]
implies
X=Y
```

3. Knowledge Type 3: A Property Causes an Event In this category, if a property P causes an event E then an entity that participates in E is also associated with P. Formally, the generalized representation of this category is as follows.

```
P [instance_of: PROP, is_trait_of: X_1,..., is_trait_of: X_m]
causes
E [instance_of: A, rel_1: Y_1,..., rel_n: Y_n]
implies
X=Y
```

where, $X \in \bigcup_{i=1}^{m} \{X_i\}$, and $Y \in \bigcup_{j=1}^{n} \{Y_j\}$

Intuitively, a statement of the above category means that if the execution of an event (E) is caused by a property (P) then an entity that is associated with P is also a participant in E.
A winograd schema example that requires the knowledge from above category is mentioned below.

Example 1:

Sentence: The sculpture rolled off the shelf because it wasn't anchored.
Question: What wasn't anchored?
Answers: The sculpture.
Knowledge Needed:

```
P [instance_of: not anchored,
   is_trait_of: X [instance-of:entity]]
causes
E [instance_of: roll off
       recipient: Y_1 [instance-of: entity]]
implies
X=Y_1
```

4. Knowledge Type 4: An Event Causes a Property In this category, if an event E causes a property P then an entity that participates in E is also associated with P. Formally, the generalized representation of this category is as follows.

```
E [instance_of: A, rel_1: X_1,..., rel_n: X_m]
causes
P [instance_of: PROP, is_trait_of: Y_1,..., is_trait_of: Y_n]
implies
X=Y
```

where, $X \in \bigcup_{i=1}^{m} \{X_i\}$, and $Y \in \bigcup_{j=1}^{n} \{Y_j\}$

Intuitively, a statement of the above category means that if the execution of an event (E) causes a property (P) then an entity that participates in E is also associated with P. The winograd schema example below explains the above category in detail.

Example 1:

Sentence: Paul tried to call George on the phone, but he wasn't successful.
Question: Who was not successful?
Answers: Paul
Required Knowledge:

```
E [ instance_of: try_to_call;
        agent: X1 [instance_of: person]]
causes
P [ instance_of: successful,
    is_trait_of: Y [instance-of:person]]
implies
X1=Y
```

5. Knowledge Type 5: A Sequence of Events (E_{seq}) Prevents an Event (E_{n+1}) In this category of commonsense knowledge if a sequence of events E_{seq} prevents another event E_{n+1} then an entity X that participates in an event in E_{seq} also participates in the event E_{n+1}. Formally, the generalized representation of this category is as follows.

```
[+/-] E1 [instance_of: A1,
           rel11: X11,..., rel1i1: X1i1]
followed by
...
[+/-] En [instance_of: An,
           reln1: Xn1,..., relnin: Xnin]
prevents
[+] En+1 [instance_of: An+1,
           rel(n+1)1: Y(n+1)1,..., rel(n+1)j: Y(n+1)j]
implies
X = Y
```

where, $X \in \bigcup_{a=1}^{n} \bigcup_{k=1}^{i_a} \{X_{ak}\}$, and $Y \in \bigcup_{l=1}^{j} \{X_{(n+1)l}\}$

The symbol `[+/-]` in front of an event E means that E may execute or it is in-executable. In other words, the instances of the knowledge category shown by the representation above includes all the combinations of executable and in-executable versions of the events E_1 to E_n. Similarly, `[+]` in front of an event E means that E is executed and `[-]` in front of an event E means that E is in-executable (or not executed). To make the representation simpler, if an event is executable we may not add a `[+]` symbol in front of it i.e. writing `[+] E` is same as writing `E`.

Intuitively, a statement of the above category means that, if the execution of a sequent of events (E_{seq} = E_1 ... E_n) prevents an event (E_{n+1}) from executing then an entity that participates in an event in E_{seq} also participates in E_{n+1}.
The winograd schema example below explains the above category in detail.

Example 1:

Sentence: *I was trying to open the lock with the key, but someone had filled the keyhole with chewing gum, and I couldn't get it out.*
Question: *What couldn't I get out?*
Answers: *the chewing gum.*
Required Knowledge:

```
[+] E1 [instance_of: fill_with,
        recipient: X [instance_of: entity] ]
prevents
[+] E2 [instance_of: get_out,
        recipient: Y [instance_of: entity] ]
implies
X=Y
```

6. Knowledge Type 6: A Sequence of Events (E_{seq}) is Followed By an Event (E_{n+1})
In this category of commonsense knowledge if a sequence of events E_{seq} is followed by another event E_{n+1} then an entity X that participates in an event in E_{seq} also participates in the event E_{n+1}. Formally, the generalized representation of this category is as follows.

```
[+/-] E1 [instance_of: A1,
          rel11: X11, ..., rel1i1: X1i1]
followed by
...
[+/-] En [instance_of: An,
          reln1: Xn1, ..., relnin: Xnin]
followed by
[+] En+1 [instance_of: An+1,
          rel(n+1)1: X(n+1)1, ..., rel(n+1)j: X(n+1)j]
implies
X = Y
```

where, $X \in \bigcup_{a=1}^{n} \bigcup_{k=1}^{i_a} \{X_{ak}\}$, and $Y \in \bigcup_{l=1}^{j} \{X_{(n+1)l}\}$

The symbol `[+/-]` in front of an event E means that E may execute or it is in-executable. In other words, the instances of the knowledge category shown by the representation above includes all the combinations of executable and in-executable versions of the events E_1 to E_n. Similarly, `[+]` in front of an event E means that E is executed and `[-]` in front of an event E means that E is in-executable (or not executed). To make the representation simpler, if an event is executable we may not add a `[+]` symbol in front of it i.e. writing `[+] E` is same as writing `E`.

Intuitively, a statement of the above category means that if the a sequence of events (E_{seq} = E_1 ... E_n) is followed by the execution of an event (E_{n+1}) then an entity that participates in an event in E_{seq} also participates in E_{n+1}.
The winograd schema example below explains the above category in detail.

Example 1:

Sentence: *The customer walked into the bank and stabbed one of the tellers. He was immediately taken to the emergency room.*
Question: *Who was taken to the emergency room?*
Answers: *The teller.*
Required Knowledge:

```
[+] E1 [instance_of: stab,
        recipient: X [instance_of: entity]]
followed by
[+] E2 [instance_of: taken,
        agent: Y [instance_of: entity],
        destination: Z [instance_of:hospital] ]
implies
X=Y
```

7. Knowledge Type 7: An Event is followed by a Property In this category, if an event E is followed by a property P then an entity that participates in E is also associated with P. Formally, the generalized representation of this category is as follows.

```
E [instance_of: A, rel1: X1,..., reln: Xm]
followed by
P [instance_of: PROP, is_trait_of: Y1,..., is_trait_of: Yn]
implies
X=Y
```

where, $X \in \bigcup_{i=1}^{m}\{X_i\}$, and $Y \in \bigcup_{j=1}^{n}\{Y_j\}$

Intuitively, a statement of the above category means that if the execution of an event (E) is followed by a property (P) then an entity that participates in E is also associated with P.
The winograd schema example below explains the above category in detail.

Example 1:

Sentence: Sam broke both his ankles and he's walking with crutches. But a month or so from now they should be better.
Question: What should be better?

Answers: The ankles
Required Knowledge:

```
E [ instance_of: break;
        patient: X1 [instance_of: entity]]
followed by
P [ instance_of: better,
    is_trait_of: Y [instance-of:entity]]
implies
X1=Y
```

8. Knowledge Type 8: A Property is followed by an Event In this category, if a property P is followed by an event E then an entity that participates in E is also associated with P. Formally, the generalized representation of this category is as follows.

```
P [instance_of: PROP, is_trait_of: X1,..., is_trait_of: Xm]
followed by
E [instance_of: A, rel1: Y1,..., reln: Yn]
implies
X=Y
```

where, $X \in \bigcup_{i=1}^{m} \{X_i\}$, and $Y \in \bigcup_{j=1}^{n} \{Y_j\}$

Intuitively, a statement of the above category means that if the execution of an event (E) is followed by a property (P) then an entity that is associated with P is also a participant in E.
A winograd schema example that requires the knowledge from above category is mentioned below.

Example 1:

Sentence: Thomson visited Cooper's grave in 1765. At that date he had been dead for five years.
Question: Who had been dead for five years?
Answers: Cooper
Knowledge Needed:

```
P [instance_of: dead,
   is_trait_of: X [instance-of:entity]]
causes
E [instance_of: visit
   co_agent: Z [instance_of: grave,
                belongs_to: Y [instance_of: entity]]]
implies
X=Y1
```

9. Knowledge Type 9: A Co-occurring Set of Events and Properties In this category, if a set of events (E_1,...,E_n) and properties (P_1,...,P_m) occur together and there is no explicit causal or sequential relationship between them, then an entity X associated with an event (E) or a property (P) in the set also participates in an event (E_y) or it is associated with a property (P_y) in the set. Here, $E_y \neq$E and $P_y \neq$P. Also, if n=0 then $m > 0$ and if m=0 then $n > 0$. Formally, the generalized representation of this category is as follows.

```
E1 [instance_of: A1, rel11: Y11,..., rel1k: Y1k]
and
...
and
En [instance_of: An, reln1: Yn1,..., relnl: Ynl]
and
P1 [instance_of: PROP, is_trait_of: X11,..., is_trait_of: X1i]
and
...
and
Pm [instance_of: PROP, is_trait_of: Xm1,..., is_trait_of: Xmj]
implies
X=Y
```

where, X,Y $\in \mathcal{S}_1 \cup \mathcal{S}_2$, such that $\mathcal{S}_1$ is a set of all the entities that participate in events $E_1,...,E_n$ and $\mathcal{S}_2$ is a set of all the entities that are associated with the properties $P_1,...,P_m$. Also, X and Y are associated with different events/properties.

Intuitively, a statement of the above category means that if a set (S) of events and properties exist together then an entity that is associated with an event E or property P in S is also associated with a different event/property in S.

A winograd schema example that requires the knowledge from above category is mentioned below.

Example 1:

Sentence: The fish ate the worm. It was hungry.
Question: What was hungry?
Answers: Fish
Knowledge Needed:

```
E [instance_of: eat
   agent: X [instance_of: entity]]
and
P [instance_of: hungry,
   is_trait_of: Y [instance-of:entity]]
implies
X=Y
```

10. Knowledge Type 10: A Set of Properties Causes a Property In this category, if a set of properties (P_1,...,P_m) occurring together causes another property (P) then an entity X associated with a property (P) in the set is also associated with the property P. Formally, the generalized representation of this category is as follows.

```
P₁ [instance_of: PROP₁, is_trait_of: X₁₁,..., is_trait_of: X₁ᵢ]
and
...
and
Pₘ [instance_of: PROPₘ, is_trait_of: Xₘ₁,..., is_trait_of: Xₘⱼ]
causes
P [instance_of: PROP, is_trait_of: Y₁,..., is_trait_of: Yₖ]
implies
X=Y
```

where, X $\in \mathcal{S}_1$, such that $\mathcal{S}_1$ is a set of all the entities that are associated with the properties $\mathtt{P}_1, \ldots, \mathtt{P}_m$ and Y $\in \bigcup_{i=1}^{k} \{Y_i\}$.
Intuitively, a statement of the above category means that if a set (S) of properties, existing together, causes another property P then an entity that is associated with a property P_i in S is also associated with P.
A winograd schema example that requires the knowledge from above category is mentioned below.

Example 1:

Sentence: Sam and Amy are passionately in love, but Amy's parents are unhappy about it, because they are fifteen.
Question: Who are fifteen?
Answers: Sam and Amy
Knowledge Needed:

```
P₁ [instance_of: in_love,
    is_trait_of: Y [instance_of: entity]]
and
P₂ [instance_of: fifteen years old,
    is_trait_of: X [instance_of: entity]]
causes
P [instance_of: unhappy,
   is_trait_of: Z [instance_of: entity,
                   instance_of: parents,
                   related_to: Y [instance_of: entity]]]
implies
X=Y
```

11. Knowledge Type 11: Specific Events, Properties and Entities This type consists of relationships between specific events, properties and entities. There is no generalized representation for this kind of knowledge. An example of this kind is as shown below.

Example 1:

Sentence: Fred is the only man alive who still remembers my father as an infant. When Fred first saw my father, he was twelve years old.
Question: Who was twelve years old?
Answers: Fred
Knowledge Needed:

`a twelve years old is not an infant`

Here, exists a specific event (`not being`) with respect to specific properties `twelve year old` and `infant`

12. Knowledge Type 12: Specific Events, Properties and Entities This type consists of a comparative statement between the existence probabilities of two different propositions. There is no generalized representation for this kind of knowledge. An example of this kind is as shown below.

For example, Prob(Sheep looks like a dog) > Prob(Shepherd looks like a dog)

Example 1:

Sentence: Sam tried to paint a picture of shepherds with sheep, but they ended up looking more like dogs.
Question: What looked like dogs?
Answers: The sheep
Knowledge Needed:

`P(sheep look like dogs) > P(shepherds look like dogs)`

The above knowledge instance represents that the proposition `Sheep look like dogs` is more probable than the proposition `shepherds look like dogs`