A Probabilistic-Logic based Commonsense Representation Framework for Modelling Inferences with Multiple Antecedents and Varying Likelihoods

Shantanu Jaiswal, Liu Yan, Dongkyu Choi, Kenneth Kwok

Social and Cognitive Computing Department, Institute of High Performance Computing, A*STAR, Singapore

Abstract

Commonsense knowledge-graphs (CKGs) are important resources towards building machines that can ‘reason’ on text or environmental inputs and make inferences beyond perception. While current CKGs encode world knowledge for a large number of concepts and have been effectively utilized for incorporating commonsense in neural models, they primarily encode declarative or single-condition inferential knowledge and assume all conceptual beliefs to have the same likelihood. Further, these CKGs utilize a limited set of relations shared across concepts and lack a coherent knowledge organization structure resulting in redundancies as well as sparsity across the larger knowledge graph. Consequently, today’s CKGs, while useful for a first level of reasoning, do not adequately capture deeper human-level commonsense inferences which can be more nuanced and influenced by multiple contextual or situational factors.

Accordingly, in this work, we study how commonsense knowledge can be better represented by – (i) utilizing a probabilistic logic representation scheme to model composite inferential knowledge and represent conceptual beliefs with varying likelihoods, and (ii) incorporating a hierarchical conceptual ontology to identify salient concept-relevant relations and organize beliefs at different conceptual levels. Our resulting knowledge representation framework can encode a wider variety of world knowledge and represent beliefs flexibly using grounded concepts as well as free-text phrases. As a result, the framework can be utilized as both a traditional free-text knowledge graph and a grounded logic-based inference system more suitable for neuro-symbolic applications. We describe how we extend the PrimeNet knowledge base with our framework through crowd-sourcing and expert-annotation, and demonstrate how it can be applied for deeper and more interpretable passage-based semantic parsing and question answering.

Keywords: Commonsense representation and reasoning, Probabilistic logic programming

1. Introduction

The development of commonsense knowledge resources has been a prominent line of research in enabling artificial intelligence (AI) models to not only interpret text or environmental scenes, but to also derive inferences upon them to support higher-level cognitive tasks [1, 2]. These knowledge-bases are typically designed to capture knowledge of the world corresponding to the mental model of an average human that is often implicit [3, 4, 5] and not directly learnable from text or images alone.

While the first large-scale commonsense resources such as Cyc [6] utilized specialized internal representations (e.g. in CycL) to encode knowledge, more recent resources such as ConceptNet [7, 8] and ATOMIC [9, 10] encode knowledge as free-text tuples of form (head, relation, tail) as part of a larger knowledge graph. The relative simplicity of the latter representation scheme has resulted in them being effective resources for integrating commonsense in neural language [11, 12, 13, 14] and vision models [15, 16, 17, 18], designing appropriate ‘reasoning’ benchmarks [19, 20, 21] and being utilized to enhance ‘reasoning abilities’ of large-scale language models [9, 22, 23].

However, a prominent limitation of the free-text (head, relation, tail) representation scheme is that the knowledge it can express remains limited to declarative facts as in ConceptNet [8] (e.g. ‘human’, ‘capable_of’, ‘buying things’) or single-condition inferential knowledge as in ATOMIC (e.g. ‘X eats Y’, ‘has_subevent’ ‘X chews Y’). Such knowledge although useful at a surface level, only constitutes a portion of human commonsense which is often more nuanced and contextually-informed. For example, current CKBs cannot encode conceptual dependencies such as “a human can
buy an object only if the money or assets they possess are greater than or equal to the object’s cost. Further, the subevent of X eats Y is much more likely to be X drinks Y (and much less likely to be X chews Y)\(^1\). Consequently, the tuple representation scheme does not encode likelihoods to differentiate certainties in knowledge. As shown in fig. 1 beliefs such as \(('human', 'desires', 'car')\) or \(('X eats Y', 'subevent', 'X chews Y')\), that should be more tentative, are treated with the same likelihood to beliefs such as \(('human', 'has', 'brain')\) or \(('X eats Y', 'causes', 'X gains energy')\), that should presumably be more certain.

**Need for probabilistic-logic based representations of commonsense.** One may look to address these limitations by simply representing conceptual or contextual antecedents in a single-text phrase (e.g. \('([X \text{ eats } Y] \text{ and } [Y \text{ is liquid}], \text{ 'subevent'}, \text{ 'X drinks Y'})\) and by adding a likelihood variable (e.g. \('([\text{'human'}, \text{ 'capable_of'}, \text{ 'buying things'}], 0.6)\)). However, this still does not effectively capture the composite nature of such knowledge wherein the inference and its likelihood is derived dynamically from a logical computation of relevant antecedents. Hence, for a more comprehensive representation scheme, we propose the utilization of probabilistic logic programming \([24]\) wherein, as illustrated in fig. 1 base or context-independent conceptual beliefs can be represented as probabilistic facts (e.g. 0.95::desires(person, car)) and composite or contextually-derived beliefs can be represented as probabilistic clauses (e.g. 0.95::capable_of(X, buy, Z): haas(X, Y) & value(Y)>=value(Z) which states that \('X \text{ can buy } Z \text{ if } X \text{ haas } Y \text{ and value of } Y \text{ is more than equal } Z \text{ 's value').

Identifying salient relations and structuring knowledge with a hierarchical conceptual ontology. Another limitation of prominent CKGs is that they utilize a small list of relations shared across all concepts (e.g. ConceptNet (v5.7) \([8]\) has 36 relations and ATOMIC-2020 \([9]\) has 11). As noted by previous work \([9, 23]\), this effectively limits the variety of commonsense captured by these CKGs resulting in them missing out on nuanced concept knowledge such as affordances, social and situational properties that form a vital portion of human commonsense and are also more difficult to capture through large-scale pretraining. Further, these CKGs lack a coherent knowledge organization structure and collect knowledge for a concept by only looking at the head phrase (of a tuple) in isolation without considering previously collected knowledge for related or parent concepts. This results in redundancies and sparsity across the larger knowledge graph.

To address these limitations, we propose the incorporation of a multi-level hierarchical conceptual ontology inspired by past cognitive models of human concept categorization \([25, 26]\). As shown in fig. 2 such a hierarchical ontology proceeds from broad ontological classes (e.g. REAL, MANMADE, PHYSICAL, etc) to relatively more specific “conceptual groups” \([27]\) (e.g. VEHICLE, PERSON, etc) from which finally actual concepts are derived (e.g. Car – [Car, Aeroplane, ..]). As shown in table 1 each concept node are attached relations and base beliefs that are inherited by their children nodes (unless over-ridden). This enables relations to be identified specific to conceptual-nodes (rather than globally for the entire knowledge graph) and to be progressively inherited downwards (e.g. ‘Vehicle’ has specific relations [mileage, top_speed, travel_area] and also inherits relations from parents such as ‘Physical’: [size, location, etc] and ‘Manmade’: [usage, construction, etc]). Consequently, two lower-level concepts can have different applicable relations depending on their parents (e.g. ‘Vehicle’ shares ‘Physical’ relations with ‘Programmer’ but not ‘Vehicle’ relations).
Similarly, conceptual beliefs (facts/clauses) can also be encoded at different conceptual levels and inherited downwards unless over-ridden (e.g. “(X, can, think)” and “(X, has, feelings)” is inherited by ‘Programmer’ node from its parent ‘Sentient’ and more specific beliefs such as “(programmer, uses, terminal)” are specified at the ‘Programmer’ node). This effectively promotes re-use of knowledge when adding new lower-level concepts (similar to human few-shot concept learning) and helps better identify concept-specific beliefs with lesser resultant redundancy across the knowledge-graph.

**Probabilistic-logic based commonsense representation framework.** Based on the above two insights of utilizing a probabilistic-logic knowledge representation scheme and incorporating a hierarchical conceptual ontology, we propose a probabilistic-logic based commonsense representation framework. The framework is designed to model a wide variety of world knowledge and composite inferences including temporal verb-schemas, situational beliefs, physics laws and higher-order beliefs (as detailed in sec. 3). Its representation scheme is flexible such that knowledge can be represented through logical combinations of both grounded concepts and free-text phrases (as shown in fig. 1 for subevent “X eats Y”, “chew” and its antecedents – e.g. “malleable” is free-text while ‘solid’ and ‘meat’ are grounded concepts). In effect, our framework builds upon existing free-text phrase representation schemes and can be used to enrich existing CKGs with a further variety of knowledge and reasoning capabilities. In this work, we specifically extend the PrimeNet knowledge-base [28] with our framework, and demonstrate its usage as a grounded logic-based inference system more suitable for neuro-symbolic applications.

The rest of the paper is organized as follows – (i) first, we review relevant work, (ii) second, we describe ProbNet in detail and how we construct a first version of it through both crowdsourcing and manual annotation, (iii) third, we illustrate its inference engine and how it can be applied for deeper and more interpretable passage-based semantic parsing and question answering, and (iv) finally, we discuss limitations of our current work and future directions.

### 2. Related work

2.1. **Commonsense knowledge resources and graphs**

Initial works in creation of machine-readable commonsense resources include the Cyc knowledge base [6] and the Open Mind Commonsense Sense project [29] which evolved into ConceptNet[7, 8]. While Cyc attempted to hand-code commonsense knowledge in specialized ‘CycL’ representations, ConceptNet obtained
commonsense knowledge from the general public in text-phrase tuple forms through appropriate crowd-sourcing mechanisms. Given its larger coverage and relatively simple representation scheme, ConceptNet has found greater utilization with recent neural (deep learning) approaches to improve ‘reasoning’ abilities. Other prominent knowledge resources have also utilized relatively simpler representation schemes than Cyc, resulting in them being able to encode knowledge at a large-scale and be utilized with neural methods. Prominent examples include SenticNet [30] for modelling knowledge relevant to sentiment analysis tasks, WebChild [31] and ASER knowledge graph [32] that extract information through web crawling, ATOMIC [10, 9] which models ‘if-then’ inferential knowledge for event situations and CSKG [33] that encodes knowledge across different such resources to obtain a unified knowledge graph. For a more comprehensive survey, we refer the reader to [34].

In relation to these works, our work focuses on how knowledge can be better represented by utilizing probabilistic logic to model composite and dynamic inferences, and incorporating a hierarchical ontology to structure beliefs and identify salient relations. We believe this can be a complementary direction to the development of further commonsense resources or updates in existing ones, particularly given the growing developments in neuro-symbolic learning methods [35, 36, 37] and their utilization for reasoning tasks [38, 39, 40].

2.2. Commonsense datasets and neural knowledge graphs

An alternative direction to developing commonsense knowledge resources in form of knowledge-graphs or -bases has been to curate datasets for specific types of commonsense reasoning. These datasets can then be used to train neural or large-scale pretrained models to better capture the targeted type of commonsense, besides serving as benchmarks for the same. Some prominent works in this direction include Piqa [41] for physical commonsense, VCR for visual commonsense [42], CICERO [43] for contextually-informed dialogue inferences, SocialQA [44] for social situational commonsense and TIMEDIAL [45] for temporal commonsense for dialogue inferences. Further, works [9] have also looked into how commonsense resources can be combined with large-scale pretrained language models (see [23] for an extensive survey) to be represented as ‘neural knowledge graphs’ and queryable through prompting. In relation to the above directions, we believe our work can support development of newer datasets / benchmarks that evaluate composite logical processing (similar to how CommonsenseQA [20] utilizes ConceptNet) besides being utilized to potentially enable neural knowledge graphs to make contextually-dependent and logical inferences with varying likelihoods / certainties.

2.3. Answer set programming frameworks

Answer set programming (ASP) [46, 47] is another prominent logic-based knowledge representation and reasoning approach. In ASP, given knowledge (rules and facts), valid models (specifically ‘stable Herbrand models’ that contain no variables) are generated from which solutions can be inferred. This is a bottom-up model generation approach in contrast to traditional Prolog-like logic programming wherein given a query, backward chaining is performed with matching rules/facts to prove (or disprove) the query. Recent works have shown how ASP can be utilized for natural language inference tasks [48]. However, in contrast to probabilistic-logic frameworks such as ProbLog, prominent ASP frameworks do not allow for representing uncertainties in inference and lack neuro-symbolic integration, both of which motivate our choice to use probabilistic logic programming for our work.

3. Probabilistic-logic based commonsense representation framework

We aim to design a knowledge representation and reasoning scheme that can effectively model inferential knowledge comprising multiple antecedents and represent beliefs with varying likelihoods. Additionally, our target representation scheme should allow for the knowledge to be utilized as both a free-text knowledge graph and a grounded semantic inference system. Accordingly, as mentioned previously, we will utilize a probabilistic logic programming approach to model such knowledge and perform inferences. Specifically, we utilize the ProbLog language [49, 24] for our work given its well-developed framework [50, 51], and its extensions for relational learning and neuro-symbolic processing [39], which are both relevant to future directions of our work on learning such knowledge.

3.1. Probabilistic logic programming primer / syntax

Probabilistic logic programming extends traditional logic programming [52] (as in Prolog) with probabilities attached to facts and clauses to model uncertainties. Table A.11 (appendix) lists probabilistic logic programming terminology and ProbLog syntax relevant to our work’s discussion. An example representation of the natural language belief ‘X
3.2. Extending PrimeNet with a base conceptual ontology and typed predicates

PrimeNet is a commonsense knowledge base organised hierarchically around a basic level of concepts, in terms of which humans tend to think about the world. It was constructed by organising these basic level concepts under a superordinate level of much fewer primitive classes (hence the name PrimeNet), and over a subordinate level of many more entity classes which are specialisations of the basic level concepts. PrimeNet is psychologically inspired from observations that humans rely on a much smaller set of concepts to function in the world compared to the millions of lexical concepts that exist, and hypothesises that commonsense reasoning could similarly depend on a concise core of primitive concepts as shown in fig. 3.

Our commonsense knowledge representation and reasoning scheme extends PrimeNet as follows:

1. We ground the primitive level classes in PrimeNet to a base ontology adapted from Dahlgren [27] as previously introduced in fig 2. We do this by equating conceptual primitives in PrimeNet to Kind Types in the base ontology.
2. We introduce type-specific predicates (relations and attributes) to each concept node that are inherited downwards by subordinate concepts.
3. We associate probabilities with each conceptual belief (facts/clauses) to reflect the likelihood that people would hold such a belief.

Our ontology’s hierarchy was designed specifically to comprise a lesser number of nodes and levels in comparison to the more extensive WordNet hierarchy [53]. This was done as our aim was to utilize the hierarchy to identify concept-salient predicates and encode beliefs at different levels in a manageable manner. A subset of the ontology with salient predicates is shown in table 3.2. With such a structure, new predicates, beliefs and concepts can be added incrementally at lower-levels to gradually increase the variety of knowledge expressible. For events, as we detail later, in addition to conceptual predicates such as ‘causes’, ‘enables’, etc., we utilize semantic roles drawn from VerbNet [54] to represent particular instances of events and any beliefs pertaining to entities.
3.3. Knowledge representation and reasoning scheme

To illustrate our knowledge representation and reasoning scheme, we make use of the format as in table [3] where ‘Knowledge’ refers to encoded facts and beliefs. ‘Example queries’ indicate inference or retrieval-time queries and ‘Inferences’ indicate their corresponding results. Note, that knowledge indicated in further examples are to illustrate the representation scheme capabilities with familiar concepts / examples and not all are necessarily encoded in the first version of ProbNet’s knowledge-base version (which is constructed for aerospace concepts as detailed in sec. [4]).

| Knowledge |
|---|
| F1 0.6::can(animal, move). |
| F2 0.8::has(car, wheel). |
| C1 0.9::can(X, move); has(X, leg); has(X, wheel). |

Table 3: Example format used to illustrate representation scheme and inferences (F1 and F2 are distinct facts, C1 is a clause, Q1 and Q2 are distinct queries while I1 and I2 are their corresponding inferences).

General representation scheme. We represent declarative facts in the following format:

\[ P: head\_predicate([fact\_id], [source\_id], time\_point, pred\_arg1, pred\_arg2, \ldots). \]

Here \( P \) refers to probability of fact, \([fact\_id]\) refers to a unique fact identifier, \([source\_id]\) refers to source of fact, \( time\_point \) indicates the time at which a fact is valid and the arguments thereafter are the original predicate specific arguments. While \( P \) of a fact can be a continuous number between 0 to 1, in this work we utilize four discrete ‘certainty’ levels with corresponding \( P \). These are – (i) tentative (\( P=0.5 \)), (ii) likely (\( P=0.7 \)), (iii) strongly likely (\( P=0.9 \)), and (iv) inherently true (\( P=1.0 \)).

The fact and source identifiers are both used for interpretability when multiple-hop inferences are performed with different knowledge sources (detailed later). While one could achieve inference trace through just the \( fact\_id \), we maintain a distinct source argument to enable source-dependent logic (e.g. some knowledge sources might be known to be unreliable and thereby be given low likelihood). Further, the trace retrieval can also be used to differentiate cases wherein a query is evaluated False (or likelihood=0) due to absence of any matching knowledge (in which case trace is empty) or due to a rule/fact activation (in which case trace is not empty). This effectively relaxes the closed-world processing assumption.

The time point indicator is introduced for temporal and event logic (detailed later), and the default time point is \( t\_g \) denoting general time (indicating a fact is assumed to be generally valid irrespective of time unless over-ridden for a particular time point).

We represent inference clauses with \( N \) antecedents in the following format:

\[ P: head\_predicate(Fh,Sh,Th, args):- \]

\[ \text{anc1\_pred}(F1,S1,T1,a1\_args...), \ldots \]

\[ \text{ancN\_pred}(FN,SN,TN,aN\_args...), \]

\[ \text{union\_ops}(Fh, C\_id, F1 \ldots FN), \]

\[ \text{union\_ops}(Sh, S1 \ldots SN). \]

The latter two ‘union\_ops’ refer to list operations (detailed in [4] through Prolog operators ‘append’ and ‘sort’) to derive \( Fh \) and \( Sh \) through union of \( F1 \) to \( FN \) and \( S1 \) to \( SN \) respectively. \( C\_id \) refers to the unique clause identifier which is added for interpretability trace.

| Knowledge |
|---|
| F1 1::isa([f1], ['wnt'], Tg, person, organism). |
| F2 1::isa([f2], ['kb'], Tg, programmer, person). |
| F3 0::isa([f3], ['kb'], Tg, person, car). |

\%Example clause with trace

\[ C1::\text{isa}(F,S,T,X,Y)\rightarrow \text{isa}(F1,S1,T,X,Y), \text{append}([S1,S2], S3), \text{sort}(S3, S), \text{append}([F1, F2], F3), \text{sort}([C1|F3], F). \]

Table 4: Base representation scheme with inference traces. As shown for inference I4, the 1st derivation (Y=person) is direct fact lookup (F2), while the 2nd derivation (Y=organism) applies clause C1 on F2 and then finds F1. Further, traces relax closed-world assumption as shown for I2 where isa(person,programmer) is indicated unknown, cf. I3 where isa(person,car) is known from F3 to be false.

An example utilization of the above scheme is shown in table [4]. Two basic facts of isa(person, organism) and isa(programmer, person) are encoded with the first obtained from source ‘wnet’ and second from
source ‘kb’. An inference rule for inheritance is represented as shown in C1 (the basic rule being isa(X,Z):- isa(X,Y),isa(Y,Z)), and augmented with list operations for trace encoding. The first query Q1 is solved through direct retrieval of F1, the second query Q2 is indicated to be unknown as no matching facts were found and the third query Q3 is indicated to be False based on fact F3. Finally, Q4 which asks what all things (variable ‘Y’) a programmer is, is resolved through retrieval of F2 and then application of C1 and retrieval of F1.

For the remainder of our paper, we omit indication of variables F, S and T in facts or clauses unless relevant.

Hierarchical knowledge inheritance. As shown in table 5 beliefs can be attached at different levels (e.g. an ‘animal’ can do a ‘motor action’ is represented in F1, while ‘bird’ can ‘fly’ is represented in F7). To inherit beliefs (or any predicate in general), we utilize the inheritance rules listed in C1 and C2. While C1 specifies that “X is Z if X is Y and Y is Z”, C2 specifies that “if X is Z and Z can do Y, then X can do Y provided that no knowledge that X cannot do Y is mentioned”. As shown in Q1, the belief that a ‘sparrow’ can do a motion activity is inferred through inheritance of properties from ‘sparrow’s’ parent node ‘animal’ through application of C2 and C1. Similarly as shown in Q2, ‘duck’ inherits beliefs that it can fly from parent ‘bird’ and in Q6, it has its own concept-level belief that ‘duck’ can ‘swim’ (which is unknown for a ‘sparrow’ as shown in Q7). While shown for the predicate ‘can’, such inheritance can be applied with all existing predicates through the same rule-type as in C2.

To specify that a belief is not true, the ‘not’ prefix is assigned to the existing predicate. As shown for F13, it is used to specify that a ‘penguin cannot fly’. Similarly, in F15, for a hypothetical ‘new_bird’, the inherited belief that ‘X can fly’ is made tentative by adding the ‘not’ prefix. One may attempt to override knowledge by changing C2’s last antecedent to be simply not(can(X,Y)); however this will lead to a cyclic loop not permitted in ProbLog. Similarly, simply setting the likelihood to 0 as done in F14 to specify ‘penguin2 cannot fly’ will also not work since in cases wherein more than one trace exists (in this case C2 and F14), the belief is inferred through noisy-or application: \( P(f) = 1 - P(f = False) \) where \( P(f = False) = \prod_k (1 - P(trace_k)) \). As a result, for Q4 the belief that “penguin2 can fly” remains true with a probability of 0.9 (computed as 1-((1-0)*(1-0.9))). In contrast, for Q3 it is 0 as the inheritance is blocked, while for Q5, the belief “new_bird can fly” has resultant probability of 0.45 (computed as 0.9*0.5).

| Knowledge |
|-----------|
| F1 0.8::can(animal, motion_activity). |
| F2 isa(bird, animal). |
| F3-4 isa(fly, motion_activity). isa(swim, motion_activity). |
| F7 0.9::can(bird, fly). |
| F8-10 isa(sparrow, bird). isa(duck, bird), isa(penguin, bird). |
| F11 isa(penguin2, bird). |
| F12 isa(new_bird, bird). |
| F13 not_can(penguin, fly). %Overriding inherited fact |
| F14 0::can(penguin2, fly). %Will not override |
| F15 0.5::not_can(new_bird, fly), %Override to tentative |
| F16 0.9::can(duck, swim). |
| C1 isa(X,Z):- isa(X,Y),isa(Y,Z). %basic inheritance clause |
| C2 %Below is property inheritance (unless exception) can(X,Y):- isa(X,Z),can(Z,Y), not(not_can(X,Y)). |

Table 5: Hierarchical knowledge inheritance. Clauses C1 and C2 enable beliefs to be inherited downwards such as for I1 where two-hop computation is performed. F13 overrides ‘can fly’ belief for penguin by using not_can predicate (F14 attempts to override by setting likelihood of fact to be zero but will not work due to noisy-or inference). F15 makes inherited belief ‘can fly’ for ‘new_bird’ more tentative.

Grounded concepts and free-text phrase representations. In our representation scheme, we distinguish between free-text phrases (denoted in “inverted commas”) from grounded concepts. Whereas free-text phrases are merely lexical labels, grounded concepts are concepts in our knowledge graph with associated semantic properties that can be used for inference. We illustrate this distinction by first describing how basic lookup of a concept is performed by a free-text name (lexicon lookup). As shown in table 6 F1 refers to a grounded concept bowl, which is specified as a type of container. F2 and F3 state that the concept bowl can be referred to by multiple free-text names “bowl” and “basin” (the latter with lesser probability). However, the name “bowl” can also refer to other applicable concepts, e.g. the action “roll” or “bowling” (see F7).

Querying concept beliefs can then be performed in two ways – (i) specify the actual concept (e.g. bowl) as in Q2, or (ii) specify the free-text phrase to retrieve
applicable concepts first (as in Q1, Q3, Q4) and then decide which concept to utilize (effectively performing word-sense disambiguation as in Q3 and Q4 where conditions on applicable concepts are provided). This is in contrast to CKRs wherein concepts and free-text phrases are treated as the same, which can potentially lead to duplications and inter-mixing of knowledge (e.g. beliefs pertaining to the event “bowl” or “roll” might be combined with those of the entity “bowl”).

More generally, beliefs for concepts can be encoded by using both free-text phrases and grounded concepts. As shown in F4 and F9, free-text phrases can be utilized when a grounded representation may not be yet feasible (e.g. attributes such as “eating soup” or “solid enclosure”). In C2 knowledge that X can be used to “keep things” has both a grounded antecedent (X can be close_state) and a free-text antecedent (X has a “solid enclosure”). Similarly C3 captures a free-text phrase implication with a grounded antecedent (X can “keep things” if X used for phy_storage). As a result, for Q5 which performs a free-text query (what objects can “keep things”), the concept newobj is found (due to C2) in addition to container and bowl (due to C3, C1).

Table 6: Utilization of grounded concepts and free-text phrases for knowledge representation and lexicon lookup. A concept can have multiple free-text names (e.g. in F2, F3 concept bowl has names “bowl” and “basin”) and conversely free-text names can refer to multiple concepts (e.g. free-text “bowl” refers to concepts bowl(obj) (F2) and roll(event)(F7)). Further, clauses & facts can be queried or represented using both constants & free-text (e.g. Q5 queries X that can “keep things”; similarly F4,F9 & C2.C3 for such facts & clauses).

Table 7: Event and situational inferences using semantic role knowledge. Facts F2-F4 represent base beliefs for event ‘eat’ (e.g. F2 the theme or thing eaten is probably a type of food). C1-C3 represent clauses indicating different possible subevents for ‘eat’ depending on agent or loc. of event (e.g. instrument is likely mouth if agent= dog or cat).
Event representations and situational inferences. To represent event or situational knowledge, we utilize semantic-roles [55, 56] (such as ‘theme’, ‘agent’, ‘instrument’, etc.) that we identify for broad event categories in the ontology (utilizing VerbNet [54] wherever possible). An example is illustrated in table 7 for the event ‘eat’ which is a type of ‘consume_activity’ (F1). F2 and F3 represent base context-independent beliefs, more specifically, selectional preferences [57] for the event, e.g. that the theme of ‘eat’ is a type of ‘food’ (F2), and the agent is a type of ‘animal’ (F3). F4 specifies a tentative belief that subevent of eat is ‘salivate’.

To capture contextual or situational dependencies, we make use of clauses that allow different inferences to be made on provided contextual/situational facts. For example, C1-C3 represent how the subevent of ‘eat’ can differ depending on the properties of the ‘theme’ or the thing being eaten. C1 states that the subevent is most likely chew if the theme X has physical_state = solid and that it also has property = “malleable”. C2 states that subevent is most likely “crunch” if theme X has property = “crispy” while C3 states subevent is most likely “drink” if theme X is a type of liquid.

Thus, depending on the background context information provided during querying, the inferences can vary. As shown for Q2, it is specified that the entity nachos is being eaten and that nachos are “crispy”. Consequently, the inferred subevent for this instance of eat is probably “crunch” in addition to “salivate”. Similarly, C4-C5 capture inferences for type of instrument being used for “eat” depending on agent or location of event. Q3 and Q4 illustrate two different instances of ‘eat’ both happening at a fine_dining location ‘micheli’. In Q3, agent is bob known to be a human and hence inferred instrument is probably a type of cutlery, while for Q4, the agent is a husky, and instrument is more likely to be simply mouth.

Representing event temporal implications (verb schemas). To represent event expectations or temporal implications (such as what happened before, during or after), or more formally, verb schemas [58], we utilize the time_point argument as introduced earlier. Similar to before, the temporal inferences can differ based on provided situational or semantic role knowledge for the event. As shown in table 8 C1-C4 represent temporal implications for the event enter_phy (e.g. ‘the man entered the room’). Here, the additional roles of source, destination and channel are utilized specifying where an object is originating from (source), where it is entering into (destination) and what it passes through (channel) in the process. Further, temporal attributes t_start, t_during and t_end specify the time points when an event starts, when it is going on and when it ends respectively.

C1 represents the inference that if X is a theme of E which is an instance of event enter_phy whose destination is stated to be D, then at the end of the event, the location of X is in D. Similarly, C2 and C3 capture locative implications of the theme X before or while the event is taking place. C4 additional represents the inference that an object X before entering into D, is outside D. Q1 details the scenario “air enters an engine through intake” in an appropriate logical/semantic-role representation with enter_I being an instance of enter_phy, its theme being the entity air, its destination being engine and its channel being intake. t1, t2 and t3 represent the
Physical and comparative inferences. To perform physical and comparative inferences, we build upon ProbLog’s inbuilt numerical operators (e.g. >, >=, <, etc.). As shown in table F2 and F4 represent boiling points of water (F1) and olive oil (F3) respectively. C1 denotes the ‘more than’ predicate inference which is derived by doing the appropriate relation on the value of its arguments X and Y.

Using this, physical and comparative inferences can be represented such as in C2 (“if the empty volume of X is more than size of Y, then X can contain Y”) and C3 (“if X is a liquid and its temperature is more than its boiling point, then X likely is boiling”). Q1 presents a scenario wherein ballx has size of 30cm³ and cup1 has empty volume of 20cm³, and asks whether cup1 can contain ballx. Based on C2 and C1, this evaluates to False. Similarly, Q2 represents a scenario wherein a liquid is being heated to 120 degrees celsius, and first queries the associated event for the liquid, if the liquid is water, then queries the same for if the liquid is olive oil. In the first case, the inference that associated event of water is boiling is made (due to F1, C1 and C3 – the heating temperature is higher than boiling point of water) and in the second case no inference can be made. This scheme can also support more complex knowledge such as physics laws. C4-C6 provide an example of the “ideal gas law”. C4 captures the implication of proportionality – if \(X \text{ and } Y \text{ are proportional and } Y \text{ increases}, \) or \(X \text{ and } Y \text{ are inversely proportional and } Y \text{ decreases} \) then \(X \text{ increases}. \) C5 represents similarly for \(X \text{ decreasing depending on antecedents. Finally, } C6 \text{ represents that if volume } V \text{ of a gas } X \text{ is constant, then its temperature } T \text{ and pressure } P \text{ are proportional. Further rules for other combinations of } P, V \text{ and } T \text{ can be similarly represented. } Q3 \text{ represents a scenario wherein the volume of air is known to be constant and its temperature is stated to be decreasing (which could be also implicitly provided by stating that air is a ‘theme’ of event cool). It queries whether the pressure decreases, and as shown in } I3 \text{ evaluates to True based on application of } C6 \text{ and } C5.

| Knowledge |
|------------|
| F1 | boiling_point(water,bp_water). |
| F2 | value(bp_water, 100, celsius). |
| F3 | boiling_point(olive_oil, bp_oil). |
| F4 | value(bp_oil, 300, celsius). |
| F5 | more_than(X,Y): value(X,Vx,U), value(X,Vy,U), Vx>Vy |
| F6 | can(X,contain,Y): empty_vol(X,Sx), size(Y,Sy), more_than(Sx, Sy). |
| F7 | assoc_event(X,boil):- isa(X,liquid), temp(X,T), boiling_point(X,Tbp), more_than(T,Tbp). |
| F8 | increases(X): (proportional(X,Y), increases(Y)); (invproportional(X,Y), decreases(Y)). |
| F9 | decreases(X): (invproportional(X,Y), increases(Y)); (...) |
| F10 | proportional(T,P): isa(X,gas), temp(X,T), vol(X,V), pressure(X,P), constant(V), %gas law (other combos. omitted). |

Example queries (with background knowledge):

| Inferences |
|------------|
| I1 | 0.0 can(cup1, contain, ballx) {False} |
| I2 | 1.0 assoc_event(water, E=boil). |
| I3 | 0.0 assoc_event(oil, E=?). [unknown] |
| I4 | 1.0 decreases(p1) {True} |

Table 9: Physical and comparative inferences. C1-C3 represent clauses that perform numerical comparisons to perform inferences such as whether an item can contain another object (C2) or whether a liquid will boil at a particular temperature (C3) with example inferences I1 and I2. C4-C6 encode the gas law and capture notion of proportionality/inverse proportionality with example inference I3.

Higher-order predicate inferences. Certain inferences and beliefs require higher-order logic wherein a predicate is nested in another predicate. This is especially useful for sentient type predicates (e.g. believes, desires, prefers, etc) where the second argument may
itself be a predicate/belief, e.g. believes(student, origin(universe, “big bang”)) indicating that a student believes the origin of the universe is from the “big bang”.

Similarly, as shown in Table 10, the implication of a person being a vegetarian is captured in C2. It states that if an person is vegetarian, then they would prefer eating items F (being the theme) that are not made of animal. Note, this is different from what they can physically eat from an anatomical perspective (by virtue of being an animal), which is captured in C1 (which states that if X is an animal and Y is a type of food, then X can probably eat Y). Q1 represents a scenario wherein p1 is a vegetarian, and p2 is unspecified. Different food items including kebab, tofu and pizza are indicated to be made of different items. When queried what all items p1 and p2 can eat, all items are returned as they are all types of food. However, when queried what all items p1 prefers to eat, the returned items are tofu and pizza (with 0.5 likelihood due to uncertainty of whether it is made of meat). For p2’s case however, all items are returned as possible preferences.

A similar example is also provided in C3 to encode the inference whether an agent (denoted by B) can buy an object (denoted by Z). As shown in the antecedents, this inference is true if the owner of Z (denoted by S) desires something that B owns, and also believes the value of that is more than equal to the value of Z. Q2 and Q3 also show different owners of objects, and how only p3 can buy home1 as its owner (p1) wants a car and believes that the value of p3’s car exceeds value of home1. Such a scheme could thus be potentially useful for capturing theory of mind and relevant social commonsense.

4. Knowledge-base construction and applications

In this section, we detail how we extend PrimeNet with our proposed knowledge representation scheme through crowdsourcing and manual annotation. We then illustrate its application for passage-based semantic parsing and question-answering. Our target domain for this exercise was aerospace documents and concepts (due to project funding requirements) with a focus on interpretable and ‘deeper’ semantic inferences for passage-based parsing.

4.1. Knowledge collection and analysis

A total of 574 concepts comprising original ontological nodes, nouns and verbs were identified from frequency analysis of word usage in domain text. A total of ten annotators (including the paper authors) were involved in identifying applicable concept groups within the ontology and extending the ontology with new concept groups, lower-level concepts and relevant predicates. Crowdsourcing was used to collect factual knowledge on type-specific predicates (relations and attributes) associated with each concept and to estimate their probability. Manual annotation was used for knowledge clauses and more domain-specific facts.

**Crowdsourcing setup.** 100 concepts were chosen to develop and evaluate the crowdsourcing framework. A survey-based design was used for this. Each concept was first illustrated with an example sentence to specify the word sense in which the concept is to be used. This was followed by questions to collect knowledge on relations and attributes for that concept requiring re-
Figure 4: Crowdsourced certainty levels of facts for salient concept groups. At least 3 annotators rated each fact and average score was used to filter and identify discrete certainty levels.

Free-text responses in the form of free-text phrases, or by selecting applicable options from a list (for knowledge that can be mapped to existing grounded concepts). Participants were recruited via Amazon Mechanical Turk across eight different countries. To ensure high quality of responses, participants first had to pass a word sense selection test involving 3 concepts, wherein for each concept they were shown a sentence using a particular sense of the concept and had to identify the correct sense from four possible options. Only participants who perfectly completed the selection test were permitted to proceed with the survey. Participants were then familiarized with the survey setup by doing a small practice run. For the actual survey, each participant was asked to annotate 3 concepts and each concept was assigned randomly to at least 3 annotators. Participants were given 10 minutes for each a concept, after which the survey would automatically move on to the next concept to be annotated. Example UIs for the first round of crowdsourcing are provided in appendix fig. B.7. IRB approval was obtained for the crowdsourcing exercise.

Free-text responses in the collected knowledge was first mapped to the grounded concepts in the existing knowledge-base through name retrieval and manual checking. In cases where no matching concepts were found, free-text responses were left as is. The resultant knowledge was then utilized for a second round of crowdsourcing in which responders were asked to rate a knowledge fact on a 5-point scale between \{Highly disagree, Disagree, Neutral, Agree, and Highly Agree\}. Again, at least three participants were required to rate each knowledge fact; each participant rated 30 facts. As before, participants were familiarized with the task through examples. An average score for each fact indicating possibility of it being true was computed based on mapping the responses to a range of -2 to 2 (-2 corresponding to highly disagree and 2 corresponding to highly agree). Finally, any fact having score less than equal to 0 was discarded, while scores ranging from 0.0-0.7 were counted as ‘tentative’, 0.7-1.4 as ‘likely’ and 1.4-2.0 as ‘strongly likely’. The collected responses for salient conceptual types are specified in fig. 4. As shown, a higher percentage of facts across concept types were found to be rated as likely, followed by tentative and highly likely. Concepts of type Manmade were found to have a generally lesser percentage (47.8%) of likely facts in comparison to Process and Non-physical or Measure types.

Manual annotation. To annotate clauses and more domain-specific facts, manual annotation was performed by 6 annotators having knowledge of the domain. Annotation was performed primarily to identify salient physical properties of entities, part-whole connections, event schemas, inter-event relations, entity-event relations and physics laws. Each annotated knowledge was checked by at least one other annotator for quality checking, and the final knowledge was added into the knowledge-base in form of concept facts as well as clauses.

Fig. 5 displays the resultant knowledge base statistics. As shown, it consists of 284 events, 158 real entities and 114 abstract concepts. A total of 95 predicates are used with 45 being broadly event-typed predicates, 56 being real world entity-typed and 39 being abstract-typed. Note, that predicates amongst groups overlap.
4.2. Knowledge-based semantic parsing and question answering

The resultant knowledge-base was applied for passage based semantic parsing and question answering. Passages were obtained from chapters of aerospace manuals in the target domain. As shown in fig. 6, here semantic parsing involved converting the input text passage into logical forms that have the same representation format as facts in our knowledge-base. The predicates for the target logical form are the same as the ones in our knowledge-base. Free-text phrases are mapped to appropriate grounded knowledge-base concepts through word-sense disambiguation (described later) or left as is (in case no mapping concept was found). For example, as shown in the figure, the free-text phrase “blockage” in the first sentence retrieves the concept obstruction_state, which is thereafter used in logic semantic forms.

**Semantic parsing module.** While one could apply a deep-learning based semantic parser [59, 60] for this task, we currently developed a rule-base parser as we did not have a significant amount of annotated logic forms corresponding to input texts to utilize as training data. We developed a rule-base that maps syntactic dependency antecedents to applicable semantic forms (with predicates from our knowledge-base). This mapping is many-to-many, i.e. a syntactic relation combination can map to multiple semantic predicates while the same semantic predicate can be activated by multiple syntactic relation combinations. Hence, it can lead to multiple possible interpretations as shown in fig. 6 wherein the first sentence “The blockage increases the pressure” is mapped to two mutually-exclusive facts cause(increase, obstruction) and agent(increase, obstruction). To resolve this, we perform selectional restriction filtering, wherein for a given event role (or more generally predicate), the applicable conceptual types for the filled argument are specified. In interpretations wherein the filled argument does not satisfy the applicable conceptual types, the interpretation is removed or its likelihood is reduced. An example is given in C1 of the same figure’s knowledge-base which states that the agent of increase event is a type of sentient. Since this fails for agent(increase, obstruction), this possible mapping is discarded. However, in the case of the second sentence, “The temperature increases with time”, for the two mutually-exclusive mappings cause(increase, time) and co-theme(increase, time), both cases satisfy selectional restriction checks and hence are stored as two distinct interpretations.

Distinct interpretations are also maintained in cases wherein multiple concepts are retrieved for a given text phrase (during word sense disambiguation) and satisfy further selectional restriction checks. Note that after finding matching concepts, an ‘instance’ of the concept is created (e.g. increase_ins1 corresponds to an instance of event increase that was specified by text-phrase “increases”, similarly, pressure_ins1 corresponds to an instance of pressure). An instance is re-used for future sentences if no determiner is provided, and a new instance for a referred concept is only created if the determiners are in \{a, another, an\}. This process is repeated for all sentences of the given passage, and the top-k interpretations are stored after each rule application (for our work, we use k=3). For each passage, the parsed forms are reset, i.e. passages are parsed independently, and we currently do not perform discourse linking.

Overall, 49 rules were identified based on domain text for mapping syntactic dependency forms to semantic forms (corresponding to knowledge-base facts based on inheritance from top-level nodes. Cumulatively across concepts, the knowledge-base encodes a total of 6799 facts (of which 2190 are for events and 2160 are for real entities) and 299 clauses (of which 198 are for events).
representations). The spacy parser [61] was utilized to obtain dependency trees for sentences. Further, for anaphora and coreference resolution, we used the “neural-coref” package within spacy, and will look into developing appropriate knowledge-based methods in future work. The resultant parser was applied on 267 sentences across 21 passages.

**Question answering module.** Given the parsed semantic forms and existing knowledge in the knowledge-base, logical queries can be directly performed as previously illustrated in sec. [3]. The different parsed interpretations are specified as different knowledge sources. Hence, when the inference is performed, the source trace specifies which interpretation was used to derive the answer in addition to the originally encoded knowledge in the knowledge-base. To allow language questions to be queried, we developed an additional template-based rule-base to convert the question into appropriate logical queries. As shown in fig. [6] the question to logical query rule-base identifies phrases matching to predicates (such as cause and to in *What causes A to X?* which are mapped to event assertions theme or agent, and the query cause(X,?)).

While some of the queries can be directly answered through the text-parsed knowledge (as for *What causes pressure to increase*?), others may require inference from both text-parsed knowledge and the knowledge-base (as in *What can decrease if the temperature is constant*?). In both cases, a trace for the answer is provided specifying which knowledge facts and clauses were utilized in answering the query, thereby providing clear interpretability in answering the question. Further, likelihoods for different possible answers can be derived based on the activated probabilistic facts or clauses, and in cases where no applicable knowledge can be found, the unknown answer is generated.

Overall, the resultant knowledge-based system allows for more interpretable semantic parsing and question answering; however, the input sentence and question for more interpretable semantic parsing and question answering; however, the input sentence and question for more interpretable semantic parsing and question answering; however, the input sentence and question for more interpretable semantic parsing and question answering.

**5. Conclusion**

We introduced a probabilistic-logic based commonsense representation framework to encode a larger variety of world knowledge and to represent conceptual beliefs with varying likelihoods. We also proposed a hierarchical conceptual ontology designed to identify salient concept relevant relations and enable beliefs to be encoded at different conceptual levels and re-used where applicable through inheritance. We illustrated the knowledge representation scheme and how it can encode different types of commonsense knowledge including contextually-dependent inferences, event schemas / temporal implications, physical and comparative inferences, and higher-order inferences. We then applied the representation scheme and ontology to extend the PrimeNet knowledge-base by crowdsourcing and manually-encoding knowledge for the aerospace domain. Finally, we illustrated how the resultant knowledge-base can be utilized for more interpretable passage-based semantic parsing and question answering.

**Acknowledgement**

This research is supported by A*STAR under its “K-EMERGE:Knowledge Extraction, Modelling and Explainable Reasoning for General Expertise” programme (Grant number A19E2b009). The crowdsourcing setup was approved by the A*STAR Institutional Review Board (IRB Reference: 2019-016).

**References**

[1] E. Davis, G. Marcus, Commonsense reasoning and commonsense knowledge in artificial intelligence, Communications of the ACM 58 (2015) 92–103.
[2] E. Cambria, A. Hussain, C. Havasi, C. Eckl, Common sense computing: From the society of mind to digital intuition and beyond, in: J. Fierrez, J. Ortega, A. Esposito, A. Drygajlo, M. Faundez-Zanuy (Eds.), Biometric ID Management and Multimodal Communication, volume 5707 of Lecture Notes in Computer Science, Springer, Berlin Heidelberg, 2009, pp. 252–259.
[3] P. J. Hayes, The naive physics manifesto, in: The Philosophy of Artificial Intelligence, 1990.
[4] H. P. Grice, Logic and conversation, in: Speech acts, Brill, 1975, pp. 41–58.
[5] S. J. Hespos, E. S. Spelke, Conceptual precursors to language, Nature 430 (2004) 453–456.
[6] D. B. Lenat, Cyc: A large-scale investment in knowledge infrastructure, Communications of the ACM 38 (1995) 33–38.
[7] H. Liu, P. Singh, Conceptnet—a practical commonsense reasoning tool-kit, BT technology journal 22 (2004) 211–226.
[8] R. Speer, J. Chin, C. Havasi, Conceptnet 5.5: An open multilingual graph of general knowledge, in: Thirty-first AAAI conference on artificial intelligence, 2017.
[9] J. D. Hwang, C. Bhagavatula, R. Le Bras, J. Da, K. Sakaguchi, A. Bosselut, Y. Choi, (comet-) atomic 2020: On symbolic and neural commonsense knowledge graphs, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, 2021, pp. 6384–6392.
Appendix A. Probabilistic logic programming primer and relevant syntax

Table A.11 lists probabilistic logic programming terminology and ProbLog syntax relevant to our work’s discussion. An example representation of the natural language belief ‘X can move with base likelihood 0.9 if X has a leg or a wheel and X is not in a stuck state’ in logical form is “0.9::can(X,move):- (has(X,leg); has(X,wheel)), not(state(X, stuck)).” For a more comprehensive overview of probabilistic logic programming, we refer the reader to [24].

Appendix B. Crowdsourcing annotation UI

Figure B.7: Crowdsourcing round 1 screening example

Figure B.8: Crowdsourcing round 1 practice question example
| Term                | Description                                                                 | Examples                                                                 |
|---------------------|-----------------------------------------------------------------------------|-------------------------------------------------------------------------|
| **Predicate**       | A relation (e.g. isa, causes) or attribute (e.g. size, color)              | isa, shape, related_to, color..                                         |
| **Constant**        | A term that is grounded and fixed (e.g. numbers, text-phrases and symbols with lower-case first letter) | 'X eats Y', -23, person, 'Bob', x                                      |
| **Variable**        | An ungrounded term that can take values during inference.                 | X, Y, Event, Bob, Person                                                |
| **Atom**            | A predicate with n arguments; Syntax: pred(arg0,arg1..)                    | has_part(car,wheel,4), isa(X,X)                                        |
| **Logical operators** | (i) ';' denotes 'logical or', (ii) ',' denotes 'logical and' (iii) 'not' or '\+' denotes 'not' (iv) '::' denotes end of fact/clause | isa(car,vehicle), not(isa(car,apple)). isa(car,on); isa(car,off). |
| **Probabilistic fact** | A declarative belief with a given base likelihood                              | 0.5::can(x,buy,car). isa(X,X). 0.98::has(person,face). isa(person,physical). |
| **Probabilistic clause or rule** | An inferential belief where a head atom 'h' is inferred with a base clause likelihood $P_c$ from computation of body (or antecedent) set of atoms (denoted by 'b') as follows: $P(h) = P_c \times \prod P(b)$ | $P_c::$ head:- a1, (a2:a3). (Base syntax) 0.6::can(X,move):- has(X,leg). can(X,buy,Y):-has(X,Z),(value(Z)>value(Y)). |
| **Model**           | Corresponds to a set of facts and rules representing a world 'model' on which inferences can be performed | 0.5::can(item,move). 0.9::can(X,move):-has(X,wheel);has(X,leg)). |
| **Query**           | An atomic query to infer from a given world likelihood of a fact (e.g. can(human,move)?) or valid constants for variables for an atom (e.g. can(X,move)?) | can(human,move)? => 0.9 can(X,move)? => X=item (0.5); X= human(0.9) .. |

Table A.11: Terms in probabilistic logic programming and ProbLog syntax relevant to our work’s discussion.

Figure B.9: Crowdsourcing round 1 main task example

Figure B.10: Crowdsourcing round 2 main task example