Using association and containment relationships to express dependency parse trees for knowledge representation and reasoning

Fintan Nagle[1,*], Rishi Shah

[1]Department of Psychology, City St George's, London

Abstract

Abstraction is a key problem-solving technique in knowledge representation and a common representational operation allowing complex parts to be treated as simpler wholes. In combination with association (co-activation or linking via a typed relationship, as in a subject-verb-object triple), abstraction is one of the central operations cognitive agents use to build complex concepts and statements. We refer to the compositional operation which supports abstraction as "containment", arguing that it can be underprioritised in knowledge representation systems. We propose the association-containment approach: that the operations of association and containment, by which nodes may contain other knowledge graphs and may be employed as a triple's subject, object, or link, give significant representational power. We show that diverse information formats, including sentences, can be effectively represented as association-containment graphs. We apply this approach to converting dependency parse trees into association-containment form, proposing simple transformation rules which can be repeatedly applied to progressively convert dependency parse trees. These results demonstrate the flexibility of the association-containment approach and support its utility for knowledge representation.

Keywords

knowledge representation, semantic networks, knowledge graphs, parsing, abstraction, nesting

1. Introduction

How should agents represent the world in order to solve problems? We know that human and animal cognition employs a diverse variety of representational formats, such as those found in the visual object recognition pathway[1], however evidence points towards a high-level global workspace representation which carries abstract information and is accessible to consciousness and linguistic reporting[2]. Computational knowledge representation also encodes knowledge in many different formats, such as abstract syntax trees for programming languages, or triples in semantic networks.

The term "cognitive operations" has been applied to conceptual structure creation processes[3], the compositional operations necessary to build up a compound representation, such as a tree or a hierarchical labelling of the parts of an object. Two of the most important cognitive operations are identification, the ability to pick out an object from the mass of information available in the sensorium, and abstraction, the ability to combine identified objects into greater patterns.

One common meaning of abstraction, as we have seen, is as an active process: transferring a problem into a higher-level domain, identifying the commonalities and discarding the details. In this paper, we are interested in abstraction specifically as a representational operation, rather than a general problem-solving approach. We focus on the operation of containment, the representational tool which allows abstract concepts to be constructed. This distinction is useful in order to focus on the representational role of abstraction as a compositional operation in knowledge representation systems. Containment is equivalently and more informally termed "nesting".

Joint proceedings of the Workshop on Theory and Methods for Abstraction (THEMA 2026) and the Workshop on Modularity and Splitting Techniques for Knowledge Representation and Reasoning (MoST 2026) at the 9th Federated Logic Conference (FLoC 2026), July 13–29, Lisbon, Portugal

*Corresponding author.

†These authors contributed equally.

fintan.nagle@city.ac.uk (F. Nagle)

0000-0003-0514-4356 (F. Nagle); 0009-0001-3958-0052 (R. Shah)

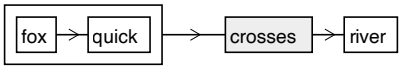
© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2026-11-15

CEUR-WS.org/Vol-4331/paper2.pdf

The following diagram shows a simple knowledge graph in which the node "fox" is directly modified by the node "quick" and this compound concept is contained, or nested, inside a box. Abstraction, in the form of containment, allows this compound concept to be used as the subject of a triple, connecting it to the object "river" through a relationship "crosses", which is shaded to indicate its role as the triple's verb:



Abstraction in the form of containment/nesting is a central component of the way we represent the world. In navigation, we are used to saying that we are inside a room, town, city, or country. We are used to placing objects inside envelopes, boxes and containers. In our interaction with computers, we are used to placing items inside folders or sub-folders and to finding tabs inside windows. All of these uses of abstraction also come with ways to move up and down the levels - to ask "what city am I in?", "what's inside this envelope", or "which folder is this file in?".

However, despite this universality of abstraction in our representation of the world, in our affordances and in the kinds of places, containers and GUIs we design, abstraction is generally given reduced priority in commonly used knowledge representation languages. We can put folders inside other folders, however, in graph databases, all nodes exist on the same level and we cannot put nodes inside other nodes. In the relational database paradigm, there is no first-class way to state that something is part of another thing; relationships are encoded indirectly by shared unique keys.

There are clear reasons to restrict the use of nesting. If a graph database, for example, were to allow nodes to be placed inside other nodes, this would require extensions to the reasoning engine and query language. Reasoning engines used in the Semantic Web make also face this tradeoff between representational flexibility and reasoning ability. The OWL Lite ontology language is decidable, but less expressive compared to the more powerful OWL DL[4]. While folders can usually be placed inside other folders, lower-level considerations such as the filesystem may intervene to prevent drives being placed inside other drives (requiring a copy to be made) or to prevent circular references.

However, cognitive agents rely on the nesting operation. Circular references are commonly found in human cognition, for example in self-consistent reinforcing beliefs[5] or the gradual self-reinforcing and selection of evidentially supported hypotheses in decision-making[6].

Containment can be made available as a representational tool to simplify the way knowledge is expressed. Document-oriented NoSQL databases such as MongoDB have rejected the join operation, a core part of the relational paradigm, in favour of storing related information in the same place using tree-structured JSON documents. In the relational world, a tree structure is not stored in the database as a tree, but as a list of nodes and a list of edges; computational work has to be done to transform it into a tree. In a MongoDB document store, the tree structure in JSON is written directly to the disk, so containers are co-located with their contents in storage. Many applications such as email clients and video editors re-implement the folder system in their own user interface because users require it for navigation.

The ubiquity of containment as a representational tool leads us to propose the association-containment view: a representational approach using just association and containment is sufficient to represent and integrate diverse types of information. In this paper, we argue in support of the association-containment view by showing that association and containment provide the tools necessary to build higher-level encodings of the information in natural language documents for the purposes of knowledge representation and reasoning.

In Section 2, we review the literature on abstraction, conceptual graphs, and information extraction from dependency parses. In Section 3, we introduce the association-containment approach. In Section 4, we describe our approach to converting dependency parse trees into association-containment form. Section 5 provides brief discussion.

2. Literature review

There is a significant amount of work on abstraction and its employment in conceptual graphs as well as on dependency parsing and information extraction from parse trees; we provide a brief review.

2.1. Abstraction

The idea of abstraction has a general definition as well as more precise connotations in particular fields. The word abstraction, from the latin abstrahere "to drag away, detach or pull away" originally referred to asceticism or a withdrawal from worldly affairs, and since the 1640s or earlier has meant an "abstract concept" or the idea of something that has no actual existence. In cognitive science, it generally means finding the common aspects between several things, or "distilling the essential".

In artificial intelligence research, specifically in knowledge representation and problem solving, abstraction generally refers to moving or transferring a problem into a higher-level domain in which the problem is simpler or faster to solve. Giunchiglia and Walsh proposed a theory of abstraction as a reasoning strategy, mapping a problem into a simpler representation in which it can be solved more efficiently[7], and describe abstraction as allowing an agent "to consider what is relevant and to forget a lot of irrelevant details which would get in the way".

The ability to abstract immediately generates separate levels of abstraction. For example, a pathfinding problem between towns could be expressed in a higher-level form on the level of countries, making a high-level solution (what countries should be visited on the way to the destination country) easier to generate; a full solution requires also solving the lower-level problems of what towns to visit on the way through each country.

Levels of abstraction are extensively employed in brain science (Marr's implementational, computational and algorithmic levels of analysis) and in electronic and software engineering (the component, gate, instruction, code, and GUI levels). In software engineering, an abstraction has the meaning of a component (such as a function or class) which solves a problem on a higher level without dealing separately with all of the possible specific cases of the problem. For example, a UserInputControl component may abstract over Buttons, DropdownMenus, and TextInputBoxes. The abstraction solves the problem on a higher level using one component, rather than using three separate components with separate code implementations.

2.2. Semantic networks and conceptual graphs

There is a large amount of existing work on semantic networks[8], knowledge graphs, and conceptual graphs[9]. The three terms are sometimes distinguished and sometimes overlap. Association is implemented universally. Abstraction is implemented to varying degrees. For example, the SNePS knowledge representation and reasoning system uses intermediate nodes to represent propositional attitudes[10].

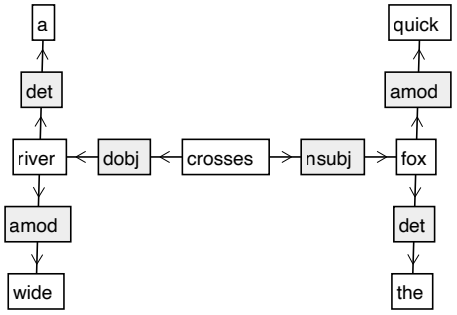
Abstraction is commonly used in everyday life to express propositional attitudes, such as in the sentence "Bob believes Alice said it is raining". The Semantic Web movement uses reification to express such higher-order statements in RDF; this requires a statement to be defined as having type Statement, then specifying the subject, predicate and object of the statement. There are several alternative approaches[11]. Yang and Kifer[12] contrast this approach against more flexible languages such as HiLog, whose variables "can range over both function and predicate symbols, [eliminating] the barrier between predicate formulas and first-order terms".

2.3. Parsing

Parsing is a knowledge representation approach derived from linguistics and specialised for the sentence level. Parsing generally results in a tree data structure. Constituency parsing uses nonterminal nodes to represent sentence components such as noun phrases. A core modern approach is dependency parsing, which links words with directed binary relations, using no intermediate nodes for phrasal

constituents. This approach was pioneered by the Stanford Dependencies project[13] and extended to multiple languages by the Universal Dependencies project[14].

The sentence "The fox crosses the wide river" has the following dependency parse tree, with the root being "crosses". The dependencies encode the direct object (dobj) and nominal subject (nsubj) of the verb "crosses", as well as equipping each noun with a determiner ("a" or "the") using the det relation and with an adjectival modifier using the amod relation. Here the dependencies are again shaded to represent their connective role.



Parsing has frequently been employed as a first step in the incorporation of information into knowledge representation systems. Information extraction systems such as KnowItAll[15] or TextRunner[16] aim to parse text from online sources and construct fact tuples. Here we aim to convert parse trees into a versatile high-level representation, the association-containment form.

3. The association-containment approach

Our approach is to consider abstraction and association as primary and ask how much representational power they make available. Association is very commonly available in knowledge representation systems. Abstraction as containment (the ability to nest composite structures inside other nodes) is less widespread, although it has been proposed[17]. Here we explore the opportunities conferred by making the ability to nest concepts a core representational operation, or "first-class citizen".

3.1. Motivation

The association-containment approach aims to extend the sentence parsing approach to allow the information in larger documents to be represented and manipulated by knowledge representation systems. Dependency or constituency parsing are specialised linguistic approaches and the trees they produce contain specialised relationships which are specific to particular linguistic theories. It is unlikely that human cognition uses exactly the same relationships as, for example, the Universal Dependency relations. Human cognition also has clear capabilities which sentence parsing does not have, such as identifying concepts across multiple sentences or representing the information present in an entire book.

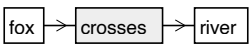
It has been suggested that the brain uses similar mechanisms to represent concepts and to represent places[18], that spatial and relational memory may be linked[19], and that space-like codes could represent word meaning[20]. This view is supported by the utility and convenience which many learners derive from constructing mind maps.

The association-containment approach begins by considering the most natural connections that can be made navigationally between a pair of places - being next to each other (association), or one being inside the other (containment) - and asks how these basic relationships can be used to represent complex knowledge structures. It has wider scope than the dependency parsing technique, being intended to allow representation of larger information structures than sentences by allowing concept nodes to be used in multiple relationships.

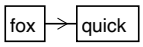
The association-containment approach also extends the Semantic Web triple by allowing any node to be used as the link in a triple. This flexibility reflects the freedom of human cognition from the need to commit to a concept being either a noun or a verb, as with "hammer". It also allows the nesting approach to be used to qualify links and allow them to encode the same complexity nodes do. In the sentence "the bird flies quickly", for example, the adverb directly modifies the verb and so should be nested inside the link node. Wherever we can employ "sprints", we should also be able to employ the compound concept "runs quickly".

3.2. Specification

As in the Semantic Web approach, we implement association using triples. A triple's link and relationship component are often limited to certain relationship types; our approach allows any node to be employed as a relationship. The following triple indicates the idea of a fox crossing a river:



When used as a relationship, a node is shaded to indicate its status as a link. Two nodes can be directly connected, or associated without using a relationship type. This models the direct co-activation of two concepts without specifying the nature of their relationship:



Like the constituency parsing approach, we use nesting to represent phrasal constituents. Each of the nouns in the previous graph can be modified by an adjective. In this case no link type is necessary for interpretation and a direct connection is used:

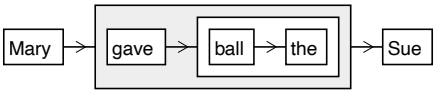


In natural language, nouns are usually equipped with determiners such as "a" or "the". Since determiners act directly on nouns, direct connections are used here. The following graph represents the sentence "the quick fox crosses a wide river":

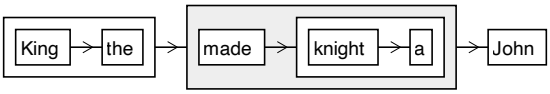


There are several key differences between the association-containment form and the dependency parse tree. The relationships in the DP tree convey specific grammatical information which is not always useful for knowledge representation and reasoning. The relationships in the association-containment graph are semantic rather than grammatical in nature. An adjective or determiner is directly connected to the noun it qualifies; a subject and object are connected via their verb.

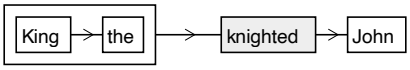
The association-containment approach provides significant flexibility when encoding ditransitive verbs, which possess multiple objects. One way to represent "Mary gave Sue the ball" is as follows:



The association-containment approach lends itself easily to transformations which move up and down the ladder of abstraction. For example, the sentence "The King made John a knight" can be represented as follows:



The triple structure allows this compound relationship to be swapped for a more precise verb with identical meaning, "knighted":



We are interested in particular in the representational flexibility available by applying nesting to the link node in a subject-verb-object triple. Our approach uses the same node type for the subject, object, and relationship, which in combination with nesting allows the relationship node to grow more complex.

3.3. Implementation

An association-containment graph is a form of semantic network. Nodes are assembled in a tree structure, with each node having a container ID. Nodes with no container ID are top-level nodes. Each node contains a list of nodes and a list of triples. Each node is assigned a randomly generated unique ID, allowing nodes to be generated in a distributed fashion.

A node may contain data; a node is a 3-tuple {ID, container_id, data}. Nodes are connected by triples, in which a node is used as the relationship; a triple is a 5-tuple {ID, container_id, subject_id, link_id, target_id} with an empty link ID indicating a direct link. Nodes and triples are either top-level or are nested inside a container node, indicated by their container ID.

Triples allow nodes to be associated in a more flexible structure than the tree structure imposed by node nesting. A triple may connect nodes which are not in the same container or on the same level of nesting. Triples can connect nodes in loops or connect nodes to themselves. Relationship nodes are not treated differently: any node may be employed as a link node and a node may participate in some triples as a subject or object and in other triples as a link. The nodes taking part in the triples of a node N do not have to be located inside N; they may be anywhere in the association-containment graph.

3.4. Summary

The association/containment approach provides a flexible and approachable alternative to parse trees, allowing information to be represented using relationships in the form of triples and reducing the importance of attaching grammatical types to relationships. Importantly, nodes can be contained (nested) inside other nodes, and any node can be employed as a relationship to connect two other nodes.

The association-containment format is able to represent diverse data structures (including trees and graphs) as well as real-world contexts such as rooms, folders, the scene grammar of images[21], the structure of documents and web pages, and the relationships present among objects and places.

We now discuss the problem of converting the information encoded in dependency parse trees to the association-containment form.

4. Converting dependency parse trees to association-containment form

We applied the association-abstraction approach to the problem of representing the information in dependency parse trees in a more appropriate form for knowledge representation and reasoning systems to work with. We show that the association-containment approach allows dependency parse trees to be naturally represented using triples. We then use a progressive approach to transform dependency parse trees into association-containment form by walking the tree and making atomic transformations. The desired end state is a knowledge graph in which all dependency relationships have been replaced with nesting or triple relationships which use words as their link. We show some examples of transformation rules and partial conversions of complex sentences.

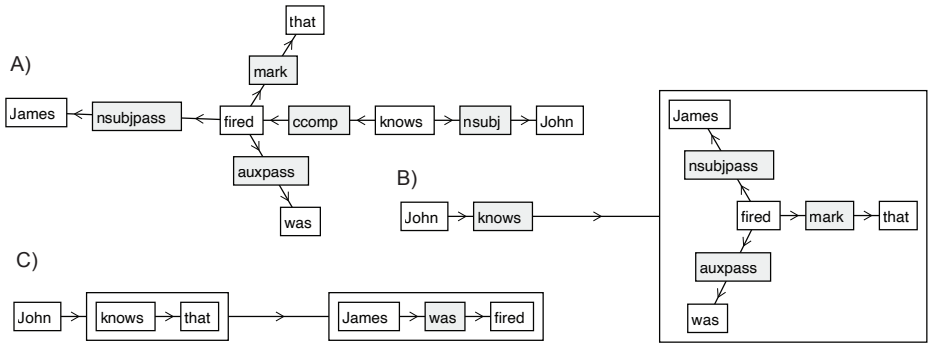


Figure 1: Transformation of the sentence "John knows that James was fired" into an association-containment graph. A) The dependency parse tree is shown, with "knows" as the root. B) The nsubj (nominal subject) relationship has been removed and replaced with a triple ("John knows..."), and the ccomp (clausal complement) relationship has been removed and replaced with a nesting operation. The clause "James has been fired" remains in its dependency parse form along with its complementiser "that". C) The remaining nsubjpass (passive nominal subject), auxpass (passive auxiliary) and mark (marker) relationships have been replaced with a triple, and the complementiser has been attached to the verb "knows", completing the transformation.

4.1. General methods

The association-containment approach was implemented in Python. To allow inspection of association-containment graphs and development of our parsing approach, a visualisation engine was developed to render graphs on screen using arrowed lines to depict association and boxes to depict containment.

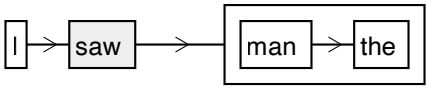
We applied our approach to a sample of 200 articles from the Simple English Wikipedia (SEW). The sample contained 4483 sentences whose parse trees contained a total of 61,731 dependencies, with a mean of 13.8 dependencies per sentence. The most common dependencies are shown in Figure 1.

Dependency parse trees were generated using the spaCy Python library for natural language processing[22] using the pretrained model `en_core_web_trf`. This model uses the dependency labels from the ClearNLP project[23]: ROOT, acl, acomp, advcl, advmod, agent, amod, appos, attr, aux, auxpass, case, cc, ccomp, compound, conj, csubj, csubjpass, dative, dep, det, dobj, expl, intj, mark, meta, neg, nmod, npadvmod, nsubj, nsubjpass, nummod, oprd, parataxis, pcomp, pobj, poss, preconj, predet, prep, prt, punct, quantmod, relcl, and xcomp.

4.2. Simple sentences

The association/containment approach allows intermediate steps in the transformation from the dependency parse tree to be easily represented in the same formalism. We aim to use transformation rules to progressively transform dependency parse trees into association-containment graphs. Our approach can be illustrated with reference to one of the most common dependency relationships, det (determiner), which is used to qualify nouns or noun phrases, as in "the man". In the association-containment representation, a determiner always requires a nested container, since the concept "the man" must be distinguished from the node "man". This is because the "man" node may play a part in other relationships elsewhere. If "the" were allowed to directly modify "man", it would modify all instances of "man" rather than just this one.

Nesting is employed most basically to represent nouns modified by adjectives or determiners. Here the amod (adjectival modifier), advmod (adverbial modifier) and det (determiner) relationships create a nested node containing a word directly modified by an adjective or determiner. The word being modified is directly connected to its modifier without requiring a relationship type:



Nesting creates a new compound node which is distinct from the original "man" node.

Generally, triples are constructed by following dobj (direct object) and nsubj (nominal subject) relationships of a dependency parse tree:



This allows a triple to be constructed using the subject and object, using the verb as the relationship:



Nesting may be employed to qualify a noun phrase with an adjective or determiner, or to encapsulate a clause, modified noun phrase, or list so that the compound structure can be employed as a triple's subject, object, or link. This is illustrated in Fig. 1, showing the transformation of a simple sentence into association-containment form.

Nesting is also employed to represent clausal constituents. The clausal complement relationships (ccomp and xcomp) and the clausal subject relationship (csubj) allow clauses (which could stand alone as sentences) to be connected in triples. Generally, clauses can be nested inside a container which is then available to be used as a subject or object.

4.3. Complex sentences

Consider the more complex sentence "A cockpit is the open space, normally near the front of an aircraft, where a pilot controls the aircraft." This sentence illustrates how natural language compresses information. It would express the same meaning if broken into separate statements: a) A cockpit is an open space; b) the pilot controls the aircraft in the cockpit; c) the cockpit is normally near the front of an aircraft. Natural language, to make the information more compact and more interesting for the reader, expresses these facts in one sentence. Once a dependency parse tree is expressed in association-containment form, transformation rules can be progressively applied to convert dependency relationships into triples (using association) and nested nodes (using abstraction) as shown in Fig. 2.

Parse trees commit to a particular interpretation of a sentence. For example, Fig. 3 shows the dependency parse tree for the sentence "Swimming has been known amongst humans since prehistoric times." This sentence uses the passive voice and could be equivalently restated as "Humans have known about swimming since prehistoric times" or "Since prehistoric times, humans have known about swimming." This allows nuance to be expressed. On the conceptual level, the sentence qualifies the act of knowing with a temporal extent ("since prehistoric times") and a direct object ("swimming").

Specifically, we could read this as using the verb "to know about swimming since" with the direct object "prehistoric times", or as using the verb "to know about something since prehistoric times" with the object "swimming". The association-containment approach allows either situation to be clearly represented. The situation can also be expressed without committing either of these concepts to the role of the direct object, by using a triple with an empty object and with both the time and activity nested inside the verb node.

Prepositional modifiers may be expressed in different ways in dependency parse trees. The sentence "The train is in London" may be parsed using a prep link (to "is") and a pobj link (to "London"), or more directly using a prep_in relationship which directly connects "train" to "London"[13]. In the first case, the transform must scan over two dependencies in order to convert to association-containment form.

4.4. Conjunctions

To convey information efficiently, natural language uses chains of conjunctions. The sentence "I like apples, oranges and lemons" has the following dependency parse tree, with "like" as the root:

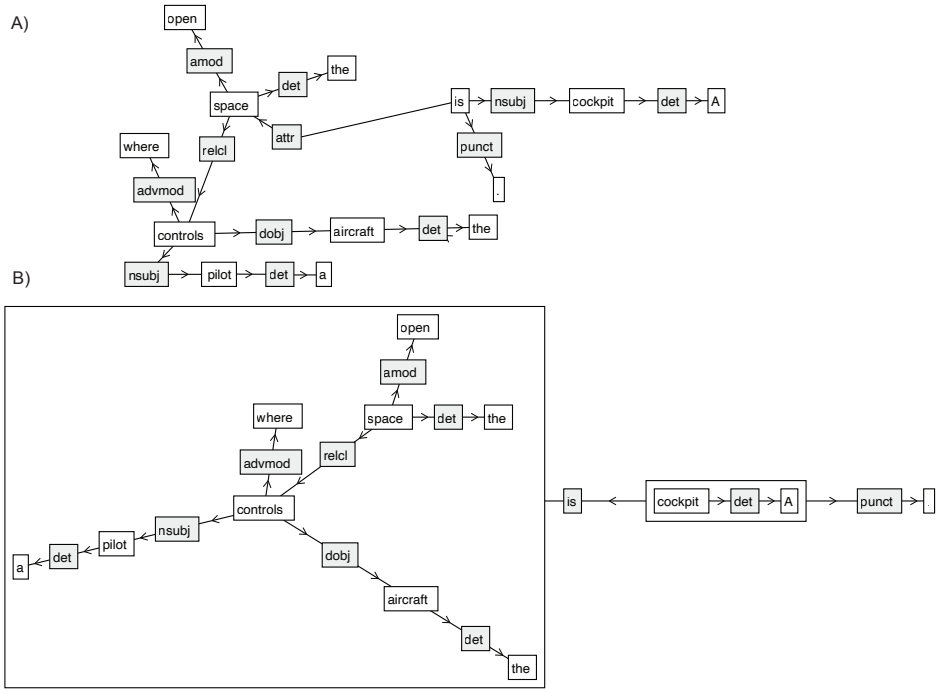
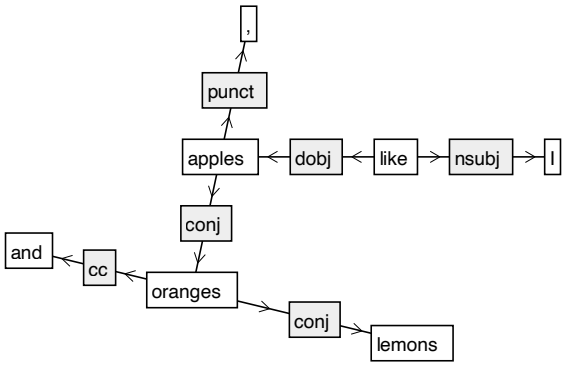
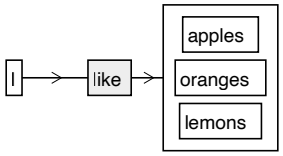


Figure 2: Partial transformation of the sentence "A cockpit is the open space, normally near the front of an aircraft, where a pilot controls the aircraft" into an association-containment graph. A) The dependency parse tree is shown, with "is" as the root. B) Firstly, the det (determiner) relationship has been applied to the "cockpit" node using nesting. The "det" node could also be removed to directly connect the determiner "A" to the "cockpit" node. Secondly, the attr (attributive) link and the nsubj (nominal subject) link, have been transformed so that the nested node containing "cockpit" is now the subject of an "is" relationship with a nested version of the remaining subtree of the dependency parse, which has been nested inside a new node. This transformation operation on the "attr" and "nsubj" links should take place atomically.



One possibility is to extract the information from this sentence as three triples: "I like apples", "I like oranges", and "I like lemons". These are naturally asserted in conjunction by being present together in the knowledge graph. The idea of "apples, oranges and lemons" can also be represented by placing the objects together in a container, employing nesting to implement the "and" relationship:



An alternative approach is to use the coordinating conjunction repeatedly as a link:

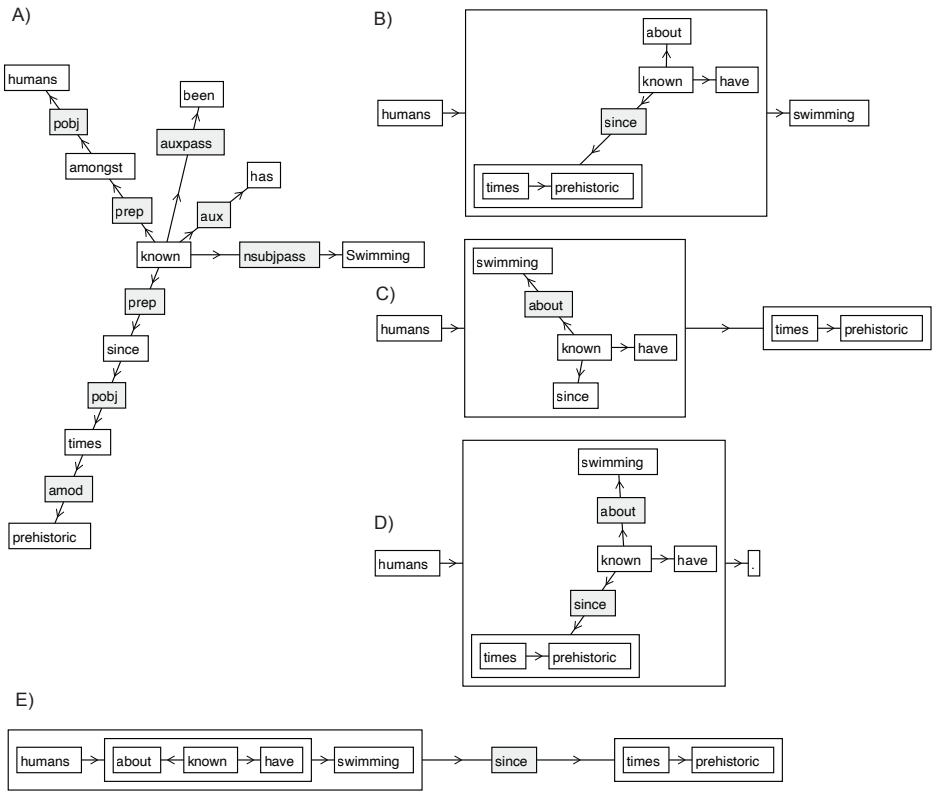
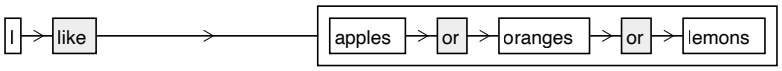


Figure 3: Transformation of the sentence "Swimming has been known amongst humans since prehistoric times" into an association-containment graph. A) The dependency parse tree is shown, with "known" as the root. This grammatical representation can be encoded in several ways. B) The temporal restriction is applied to the verb "known", with "swimming" as the object of the verb. C) The activity "swimming" now qualifies the verb "known", with the time period as the object. D) Here the object is left empty and both of the associated concepts are nested inside the relationship node. E) Alternatively, the temporal restriction can be applied to the node representing the phase "humans have known about swimming".



When extracting information from this parse tree, it is necessary to look at the "oranges" node to see that the coordinating conjunction is "and" rather than "or". Transforming a conjunction thus requires walking the parse tree until the end of the conjunction chain reveals the coordinating conjunction.

4.5. Application

We applied this approach to the 200-article Simple English Wikipedia sample, containing 4483 parse trees. Dependency parse trees were expressed in association-containment form by creating nodes for the words and the dependencies and using the dependency nodes to link the word nodes. This led to 61,731 dependencies of which the most common are shown in Table 1. A node's dependency set is the ordered, deduplicated list of its outgoing dependencies; the most common dependency sets are shown in Table 2.

Progressive conversion was then applied by performing a depth-first walk of each dependency parse tree from the root and applying the transformations described above. A single transformation was applied to each node in each tree walk; the tree walk was then iterated until no further transformations could be applied. Transformations were ordered so that, for each node, simple conversions such as that of "det" occurred first.

The approach described here allows full conversion of simple parse trees and partial conversion of

the more complex parse trees in the Simple English Wikipedia sample. During these partial conversions, 5.9 conversion operations were made on average on each parse tree.

Tag	Dependency	Freq.	Freq. %	Total %
punct	Punctuation	7512	12.17	12.17
det	Determiner	6350	10.29	22.46
pobj	Object of preposition	6345	10.28	32.73
prep	Prepositional modifier	5954	9.65	42.38
nsubj	Nominal subject	5151	8.34	50.72
amod	Adjectival modifier	3659	5.93	56.65
conj	Conjunct	3071	4.97	61.63
dobj	Direct object	2685	4.35	65.97
advmod	Adverbial modifier	2528	4.1	70.07
cc	Coordinating conjunction	2400	3.89	73.96
compound	Compound modifier	1962	3.18	77.14
aux	Auxiliary	1921	3.11	80.25

Table 1
The 12 most common dependencies in the Simple English Wikipedia sample, making up 80.25% of the 61,731 dependencies. Frequencies are also shown as percentages with a running total.

Dependency set	Freq.	Freq. %	Total %
pobj	5993	20.18	20.18
det	2271	7.65	27.83
amod	1140	3.84	31.67
prep	959	3.23	34.9
det, prep	885	2.98	37.88
amod, det	686	2.31	40.19
compound	658	2.22	42.4
attr, nsubj, punct	556	1.87	44.27
cc, conj	556	1.87	46.15
advmod	552	1.86	48.0
conj, punct	453	1.53	49.53
punct	430	1.45	50.98

Table 2
The 12 most common dependency sets in the Simple English Wikipedia sample, making up 50.98% of the 29,697 dependency sets. The tag "attr" indicates an attributive relationship[13]. Other tags are defined in Table. 1. Frequencies are also shown as percentages with a running total.

5. Discussion

Effective knowledge representation systems must be able to clearly and accurately represent facts about the world and to deal with structured collections of such facts. Here we propose the association-containment approach, the view that association and containment are sufficient for a wide variety of representational tasks.

The operations used are not new: nesting is widely used in data structures and filesystems, association via triples has a long history in knowledge representation, and intermediate nodes such as noun phrases are used in constituency parsing. The key features of this approach are the ability to nest semantic networks inside nodes and the ability to use any node as the subject, object, or relationship in a triple. We have shown that the association-containment form is sufficient to represent complex compound knowledge structures. The operations of association and abstraction allow the information carried by natural language sentences to be represented naturally and intuitively. The association-containment approach is highly flexible, allowing the encoding of self-referential concepts and loops.

We have proposed automatic generation of association-containment graphs from dependency parse trees, and presented several motivating examples showing that linguistic constructs can be effectively converted into a more abstract form using transformation rules. We have also set out a visual display formalism allowing knowledge graphs using the association-containment approach to be easily displayed and to provide rapid visual feedback on parse tree and knowledge graph structure. These results support the utility of the association-containment form and support its use as a flexible and versatile common representation format for knowledge representation and reasoning tasks.

It can be argued that employing containment is attractive simply because it allows us to see things inside other things and that this is simply a conceit of the human visual system. After all, containment can always be translated into a series of relationships in a flat (non-nested) conceptual graph. However, expressing a problem in a particular way can make solving it easier. Most programming languages express mathematical expressions using infix notation (2 + 2) rather than prefix (+ 2 2) or postfix (2 2 +) notation, because their users prefer infix notation. A clear display formalism improves a modeller's ability to iterate around the development loop. When considering the problem of converting dependency parse trees to association-containment graphs, the ability to inspect data structures during development - and to view dependency parse trees, knowledge graphs, and combinations of the two interactively - can aid development significantly. We suggest that containment is intuitive because it maps on to the cognitive containment relationships that we use to navigate our environment and which were potentially co-opted into the navigation of our information environment.

The association-containment approach enables higher-order statements such as "I think that it is raining" to be easily represented without explicit reification, variables, or named statements. Although any semantic network employing nesting can also be equivalently expressed as a flat graph, the cognitive and perceptual primacy of abstraction suggests that explicit nesting is easier to work with, both visually and conceptually.

We have shown that a wide range of sentences can be intuitively represented in association-containment form by applying simple transformations. This approach offers an intuitive and accessible way to work with dependency parse trees and to progressively convert them into higher-level formats.

Additional work remains to be done in evaluating whether the progressive conversion of dependency parse trees can be performed automatically using simple recursion or using a trained model. Even effective conversions are limited by the accuracy of dependency parsing models. The association-containment approach can be criticised for its lack of an innate approach to reasoning. Here, we have represented only individual sentences.

The association-containment form does not directly represent grammatical information but higher-level conceptual relationships. This means that, in reasoning, key distinctions such as distinguishing a determiner from an adjective must still be performed based on the words themselves rather than their grammatical roles. The flexibility of the approach also means that self-referential loops are possible and must be detected and avoided.

However, we have noted the utility of the association-containment format in representing a diverse range of knowledge structures, including raw dependency parse trees and their partial transformations. The association-containment approach is a flexible and intuitive format with considerable representational power.

Declaration on Generative AI

The author(s) have not employed any generative AI tools in any activities related to this paper.

References

[1] J. J. DiCarlo, D. D. Cox, Untangling invariant object recognition, Trends in cognitive sciences 11 (2007) 333–341.

[2] B. J. Baars, N. Geld, R. Kozma, Global workspace theory (gwt) and prefrontal cortex: recent developments, Frontiers in psychology 12 (2021) 749868.
[3] F. J. R. de Mendoza Ibáñez, Metonymy and cognitive operations, Benczes, Réka & Barcelona, Antonio & Ruiz de Mendoza Ibáñez, Fransico Jose Ruiz (eds.). Defining metonymy in cognitive linguistics: towards a consensus view. Amsterdam: John Benjamins (2011) 103–123.
[4] D. L. McGuinness, F. Van Harmelen, et al., Owl web ontology language overview, W3C recommendation 10 (2004) 2004.
[5] L. J. Rips, Circular reasoning, Cognitive science 26 (2002) 767–795.
[6] Z. Lin, C. Nie, Y. Zhang, Y. Chen, T. Yang, Evidence accumulation for value computation in the prefrontal cortex during decision making, Proceedings of the National Academy of Sciences 117 (2020) 30728–30737.
[7] F. Giunchiglia, T. Walsh, A theory of abstraction, Artificial intelligence 57 (1992) 323–389.
[8] J. F. Sowa, et al., Semantic networks, Encyclopedia of artificial intelligence 2 (1992) 1493–1511.
[9] J. F. Sowa, Conceptual graphs for representing conceptual structures, Conceptual Structures in Practice (2016) 119–154.
[10] S. C. Shapiro, An introduction to SNePS 3, in: International Conference on Conceptual Structures, Springer, 2000, pp. 510–524.
[11] D. Hernández, A. Hogan, M. Krötzsch, Reifying rdf: What works well with wikidata?, SSWS@ ISWC 1457 (2015) 32–47.
[12] G. Yang, M. Kifer, On the semantics of anonymous identity and reification, in: OTM Confederated International Conferences" On the Move to Meaningful Internet Systems", Springer, 2002, pp. 1047–1066.
[13] M.-C. De Marneffe, C. D. Manning, The stanford typed dependencies representation, in: Coling 2008: proceedings of the workshop on cross-framework and cross-domain parser evaluation, 2008, pp. 1–8.
[14] J. Nivre, D. Zeman, F. Ginter, F. Tyers, Universal Dependencies, in: A. Klementiev, L. Specia (Eds.), Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Tutorial Abstracts, Association for Computational Linguistics, Valencia, Spain, 2017. URL: https://aclanthology.org/E17-5001/.
[15] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, A. Yates, Web-scale information extraction in knowitall: (preliminary results), in: Proceedings of the 13th international conference on World Wide Web, 2004, pp. 100–110.
[16] A. Yates, M. Banko, M. Broadhead, M. J. Cafarella, O. Etzioni, S. Soderland, Textrunner: open information extraction on the web, in: Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT), 2007, pp. 25–26.
[17] A. Poulovassilis, M. Levene, A nested-graph model for the representation and manipulation of complex objects, ACM Transactions on Information Systems (TOIS) 12 (1994) 35–68.
[18] J. O'Keefe, L. Nadel, The hippocampus as a cognitive map, Oxford university press, 1978.
[19] J. C. Whittington, T. H. Muller, S. Mark, G. Chen, C. Barry, N. Burgess, T. E. Behrens, The tolman-eichenbaum machine: unifying space and relational memory through generalization in the hippocampal formation, Cell 183 (2020) 1249–1263.
[20] S. Viganò, V. Rubino, A. Di Soccio, M. Buiatti, M. Piazza, Grid-like and distance codes for representing word meaning in the human brain, NeuroImage 232 (2021) 117876.
[21] M. L.-H. Võ, S. E. Boettcher, D. Draschkow, Reading scenes: How scene grammar guides attention and aids perception in real-world environments, Current opinion in psychology 29 (2019) 205–210.
[22] M. Honnibal, I. Montani, S. Van Landeghem, A. Boyd, spaCy: Industrial-strength Natural Language Processing in Python (2020). doi:10.5281/zenodo.1212303.
[23] J. D. Choi, A. McCallum, Transition-based dependency parsing with selectional branching, in: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2013, pp. 1052–1062.