ACS Poster Collection (2018) 181–197 Published 12/2018; Paper 12

Exploring Connections Between Primitive Decomposition of Natural Language and Hierarchical Planning

Jamie C. Macbeth JMACBETH@SMITH.EDU
Department of Computer Science, Smith College, 100 Green St., Northampton, MA 01063, USA

Mark Roberts MARK.ROBERTS@NRL.NAVY.MIL
Naval Research Laboratory, 4555 Overlook Ave. SW, Washington, DC 20375, USA

Abstract

While recent research has shown that "classical" automated planning systems are effective tools for story *generation*, the success of automated story *understanding* systems may require integration between commonsense reasoning and more sophisticated forms of planning to make inferences and deductions about the plans and goals of story actors. Methods that decompose abstractions (i.e., tasks or language expressions) into primitives have played an important role for both automated planning systems and automated story understanding systems, but the two areas have remained largely isolated from each other with few overlaps. We argue that this little-explored connection can benefit both areas of research, and this position paper explores the connections between these systems through the common use of primitive decomposition and its variants. Specifically, we present a prototype of a Hierarchical Task Network planner that decomposes natural language input into primitive structures of Conceptual Dependency, a meaning representation designed for in-depth story understanding. We discuss the important challenges, implications, and applications enabled by the establishment of this unique, direct link between planning and story understanding systems.

1. Introduction

Because accounting for the plans and goals of rational agents is so important to building intelligent systems generally, it is unsurprising that aspects of planning are also important for systems that interact with users or data in the form of natural language. Recently story generation systems have successfully been integrated with automated planning systems to produce narratives or dialogues that are believable to human audiences because the characters in the narratives perform acts that align with commonsense goals and plans of readers (Riedl & Young, 2010; Haslum, 2012; Porteous et al., 2013). However, such approaches focus on sequential planning, often ignoring hierarchy, and they fail to provide a general mechanism for incorporating commonsense knowledge.

These recent successes in story generation can be applied to the problems of narrative understanding, story understanding, and ultimately language understanding. Broadly, the understanding problem can be seen as a kind of "inverse" to story generation. However, understanding is a particularly challenging task: computing systems designed to understand stories need to track the goals and plans of actors because the actions they perform are part of larger plans in the service of

higher-level goals (Wilensky, 1983). Moreover, while story generation systems can often rely on predetermined, human-generated text templates, success in language understanding and story understanding systems may require deeper integration between knowledge about goals and plans and other commonsense knowledge for decisions about parsing, resolving word sense ambiguities, and making inferences to be responsive to a broad range of inputs.

Primitive decomposition systems (Schank, 1972) take a particular approach to achieving natural language understanding by representing language semantics in terms of primitives. A unique variety of these systems transforms words, phrases, and sentences into complex structures combining non-linguistic primitives that stand for the understander's basic concepts for events, acts, and changes of state in the world. At the same time, *hierarchical planning systems* (Nau et al., 1999) express planning problems in terms of "high-level" tasks and goals, and solve them by breaking them down into intermediate subgoals and subtasks. They ultimately construct a solution as a complex sequence of primitive operators representing events, changes of state in the world, and the acts to be performed by an intelligent agent.

There has been little work examining the similarities and connections between primitive decomposition processes for natural language understanding and the decomposition processes for hierarchical planning. Exploiting these links may enhance in-depth natural language understanding systems by providing better primitive systems and decomposition strategies, and they may enhance planning systems by providing connections to natural language understanding and natural language generation systems, allowing intelligent agents to better interpret instructions and explain their actions. Additionally, these links may allow the commonsense knowledge structures present in automated planning systems be used directly for automated story understanding or vice versa.

This paper is organized as follows: Section 2 provides background on hierarchical planning systems that perform decompositions of tasks into primitive operators and story understanding systems that perform decompositions of the meanings of natural language expressions into conceptual primitives; Section 3 argues for the equivalence between these particular kinds of planning and language understanding systems and examines how their commonalities enable their integration; in Sections 4 and 5 we present an implementation and demonstration of a hierarchical task network planning system that directly decomposes a simple natural language story into a conceptualization in Schank's Conceptual Dependency system, a primitive decomposition-based meaning representation designed for automated in-depth story understanding; and in Section 6 we close with a discussion of how this prototype can be extended to include recent work in hierarchical planning and how it relates to a recent emphasis in explainable artificial intelligence.

2. Background

While our focus is on connecting automated planning to story understanding, story generation has attracted recent interest in the automated planning community, emphasizing sequential (e.g., "classical") planning systems. We provide a brief introduction to that literature and discuss work in hierarchical planning systems that perform decompositions from high-level methods to primitives. We then overview planning problems in language understanding systems and the role that decomposition into primitives plays in language understanding and story understanding.

2.1 Automated Planning in Story Generation and Dialogue Generation

Automated planning techniques have been successfully applied to dialogue systems and story generation. For example, work in dialog systems by Petrick and Foster (Petrick & Foster, 2013) demonstrated that a robotic bartender could use a variant of hierarchical planning to produce believable and interleaved dialog with multiple customers ordering beverages. Recent work has shown that knowledge of human goals and plans is important for automated story generation systems, because it makes the character actions in the texts more understandable and believable to human audiences (Riedl & Young, 2010; Haslum, 2012; Porteous & Cavazza, 2009; Porteous et al., 2013).

Automated planning systems and algorithms typically determine what actions will help intelligent agents, robots, or other autonomous systems meet their goals in the real world (i.e., in the story). Typical automated planning systems represent the world as an implicit state transition system where an intelligent agent navigates the states by applying actions that transform the world from one state to another. For example, a character in the game Minecraft desiring to move gold from the mine to a storage chest might perform the following actions: `walk_to(mine)`, `mine(gold)`, `walk_to(chest)`, and `place(gold, chest)`. The planning system takes as input an initial state (e.g., `at(home)`), a goal state (e.g., `in(chest, gold)`), and a set of action templates (i.e., operators describing the transitions), and produces a sequence of actions that transform the system from the initial state to the goal state.

Much of automated planning system research centers around designing the state abstractions, operators, and planning algorithms that construct plans while minimizing some metric (e.g., plan length, plan cost). A central focus is developing domain-independent formulations of the algorithms so that a single planning system can solve a variety of planning problems. This domain-independent approach has spawned a family of heuristics that are extremely effective at guiding the search process of the planner toward satisficing or optimal solutions. More recent advancements have extended planning to include sophisticated techniques for numeric, temporal, and continuous effects.

2.2 Decomposition in Hierarchical Planning

In contrast to constructing a sequential plan from the initial state to the goal state, hierarchical planning systems decompose abstract tasks or goals into successively more concrete action primitives. These kinds of planners are often used in industry or space applications, where planning speed is paramount and domain-dependent knowledge can greatly reduce the planning effort.

Hierarchical Task Network (HTN) planning systems such as the Simple Hierarchical Ordered Planner (SHOP) planning system (Erol et al., 1994; Nau et al., 1999, 2003) decompose abstract (i.e., compound) tasks into primitive tasks (e.g., actions). The planning process continues recursively by decomposing tasks via methods until only primitive actions remain; the resulting primitives are the plan. A challenge with HTN planning is figuring out how to provide guidance for the decomposition process. For example, when multiple decompositions exist for an abstract task, it is not always clear how the planner should choose between them. External guidance could be provided, but this increases the burden on the domain developer. Another challenge is how to deal with cases where the decomposition cannot produce a primitive or a method is missing.

A recent variant of hierarchical planning, called Hierarchical Goal Networks (HGNs, Shivashankar et al., 2012, 2013) unifies "classical" planning and hierarchical planning into a single framework. Instead of decomposing tasks, HGN planning decomposes *goals*. Problem statements contain both operators and methods, and, in the absence of a hierarchy, the planner can resort to domain-independent planning to search for a sequential plan. For both methods and operators, the planner can leverage work in heuristics from domain-independent planning to provide search guidance and reduce search effort. A stronger formal connection between HTNs and HGNs recently showed their equivalence under different semantics (Alford et al., 2016). HGNs thus blend the best of hierarchical and sequential planning approaches.

2.3 Planning in Story Understanding

Automated story understanding is a particularly challenging sub-domain of natural language understanding (Allen, 1995; Winograd, 1972). Computing systems designed to understand stories need to track the goals and plans of actors in stories because the actions they perform are part of larger plans in the service of higher-level goals (Wilensky, 1983). Story understanding tasks further raise the expectations of systems to be as knowledgeable as a human understander by resolving syntactic and semantic ambiguities in language, simulating human-like thought processes, and applying commonsense knowledge to perform inferences and fill in gaps in typical storytelling (Dyer, 1982; Winston, 2011, 2014; Diakidoy et al., 2015).

Story understanding systems perform the task of interpreting the actions of characters as described in the story by reasoning about how those actions fit into those characters' plans. A typical story understanding problem is to have a system "read" or process a story and also process and answer questions about the story (Lehnert, 1978). For example, the story could be: "Bob was thirsty. He grabbed a glass from the cupboard." And the question for the system to answer could be, "what did Bob intend to do with the glass?" The likely answer, that "Bob intended to pour a liquid into the glass and then consume the liquid," depends on the understander interpreting "Bob was thirsty" as the establishment of Bob's goal to consume liquid, which could be seen as a typical "semantics" task. However, answering the question also depends on the understander's ability to construct a sequence of actions to reaching that goal that involves the glass. Although the purpose of constructing this action sequence was to interpret the meaning of a story rather than to operate a real-world goal-directed intelligent agent, constructing that sequence of actions is identical to a planning process.

2.4 Decomposition in Story Understanding

Primitive decomposition systems are frequently used in both natural language processing research and broader areas of linguistics, because they reduce ambiguity in language by representing canonical forms of meaning. These properties of decompositions simplify implementations of natural language processing systems while enabling linguists to draw conclusions about universal properties of language (Schank, 1972; Jackendoff, 1983; Wilks & Fass, 1992; Wierzbicka, 1996).

While the majority of these primitive systems for natural language insist that the primitives must actually be words in a language, others are unique in addressing the challenges of in-depth story

understanding because they devise representations that decompose meaning into complex combinations of *language-free conceptual primitives*. Both Schank's Conceptual Dependency (Schank, 1972; Schank et al., 1975; Lytinen, 1992) and Minsky's (1988) Trans-frames seek out primitives and structures that are conceptual by virtue of being as far as possible from the language, its lexical entries, grammar, and syntax; these systems favor non-linguistic primitives representing events, acts, and changes in state in the physical and mental world of an intelligent human understander.

For story understanding, the decomposition of language into structures of language-free conceptual primitives has a number of important benefits. Firstly, representing concepts as complex combinations of primitives enables sophisticated relations between concepts and other concepts, and between concepts and language expressions (Macbeth, 2017). For example, the verb "kick" can be decomposed into two primitive acts, a MOVE primitive act representing the person moving their foot, and a PROPEL act representing the person striking an object. If the verb "punch" is similarly decomposed into a MOVE primitive act representing a person moving their fist and a PROPEL act representing a person striking an object, a natural relation between "kick" and "punch" is engendered inherently by matching the primitives within the decomposed structures. This natural relation between the concepts of "kick" and "punch" encompasses both the similarities—both mean moving a body part to strike an object—and differences—in one the foot is moved, in the other the fist is moved. This substrate also helps to overcome many of the ambiguities of surface language forms, and provides deep relations from the input text to frames, scripts, and other kinds of knowledge structures, giving language understanding systems better reasoning capabilities (Schank & Abelson, 1977). Conceptual primitive decomposition could provide a significant enhancement and complement to semantic parsing (Berant et al., 2013; Shi & Mihalcea, 2005), abstract meaning representations (Banarescu et al., 2013), and knowledgebases such as FrameNet (Baker et al., 1998) and ConceptNet (Speer & Havasi, 2013) when they are used in story understanding systems (Cambria et al., 2016, 2018).

However, one must also acknowledge the known challenges with primitive decomposition approaches generally, such as arranging for the completeness of the set of primitives, and the apparent difficulty in representing more abstract language forms. If a non-linguistic primitive representation system is chosen, another major challenge is the engineering of systems that actually perform the transformation or translation from natural language into the non-linguistic primitive decomposition representation. The primitive decomposition systems for Schank's Conceptual Dependency (CD) were often termed *conceptual analyzers* to distinguish them from parsers which only provide syntactic relations or parts of speech categories for words in texts (Riesbeck, 1975; Birnbaum & Selfridge, 1975; Dyer, 1982). Recent research in this area has connected CD primitives to image schemas and large-scale commonsense knowledge bases (Macbeth et al., 2017; Cambria et al., 2016), performed human-subject studies on the reality of conceptual primitives (Macbeth & Barionnette, 2016), and explored crowdsourcing corpora to build conceptual analyzers through machine learning (Macbeth & Grandic, 2017).

3. Primitive Decomposition in Language Understanding and Planning

Although knowledge about typical goals and plans of human actors is needed for in-depth understanding of natural language, there are many apparent differences between language understanding systems and automated planning systems. While language understanding systems work to interpret the goal-directed acts of intelligent agents or people in a text, automated planning systems determine which acts will help an agent achieve their goals.

At the same time, an intelligent agent, robot, or other autonomous system may be performing automated planning in an environment where other intelligent agents, robots, or other autonomous systems are present. These other agents—whether they are cooperative and intending to perform acts that help our agent reach its goals, or adversarial and intending to thwart our agent's goals—will be constructing plans and performing acts that will change the state of the world. To plan in this kind of environment requires the planner to be able to interpret the acts that another agent is taking and determine the ultimate goals of these acts.

Whether communication between the intelligent agents (in natural language or some sort of formal language) is possible or not, each agent will have to, in essence, *understand* the other agents by interpreting the state-changing operations and acts that they have performed in the past and by predicting what acts they will perform in the future. In a sense, the only difference between this planning process and the story understanding process described above is in the way that the planner receives its input information about the initial and goal states of the world.

In making our specific position connecting automated planning and natural language understanding, we recognize that certain deep connections between understanding and planning are well known (e.g., Wilensky, 1983; Norvig, 1983). There has even been significant work on combining the CD-transframe primitive decomposition system with representations of the goals and plans of story actors (Schank & Abelson, 1977) and using these combined representations together to build working in-depth story understanding systems (Dyer, 1982). Some recent work has focused on having robots narrate their planning experiences in natural language (Rosenthal et al., 2016), using planning systems to generate realistic narratives (Riedl & Young, 2010; Haslum, 2012), and unifying plan recognition and plan generation using lexicalized grammars (Geib, 2015). In contrast, our specific position seeks out deep connections between planning and language understanding on the basis of processes that perform decomposition into structures or networks of language-free conceptual primitives.

3.1 Primitives in Understanding and Planning: A Unified View

When one juxtaposes primitive decomposition systems for natural language understanding with hierarchical planning systems, a number of congruent structures become apparent. While both hierarchical task network and hierarchical goal network planning systems start with planning problems specified at a high level and formulate solutions in terms of low-level primitive tasks or primitive goals, primitive decomposition systems for natural language understanding transform natural language into complex structures of low-level conceptual primitives. In planning, the inputs are high-level problem statements of tasks or goals which are intuitively understandable by humans and are similar to expressions processed by natural language understanding systems (e.g. "Go to park,

good weather, no cash"). The plans that are produced are sequences of low-level task and goal operators which represent primitive acts and the consequences that those acts have on the world. These appear to be the same kinds of primitive act sequences and structures produced by a primitive decomposition of natural language.

For example, for the problem statement "Go to park, good weather, no cash", Nau et al.'s (1999) SHOP planner finds the plan `(!walk downtown park)` where the `walk` operator changes the state of the world by deleting the atom `(at ?here)` and adding the atom `(at ?there)` to the state. A primitive decomposition of "Bob walked to the park" in CD transframes (for examples, see Macbeth & Grandic, 2017) would likely produce a PTRANS act indicating that Bob traveled from his current location to the park, and a MOVE act indicating that Bob moved his legs to accomplish the PTRANS.

The method structures in hierarchical planning problems, (e.g. the `travel-to` method in Nau et al. (1999) having a task list such as `(!walk ?p ?q)` or `(!wait-for ?bus ?x) (pay-driver 1.00) (!ride ?bus ?x ?y)`) specify the decomposition process for planning, while in conceptual analysis systems for natural language, the decomposition process is defined by entries in a *conceptual lexicon*, which translates between words and phrases in natural language and CD structures, which in this case would contain PTRANS and MOVE acts. The planning system breaks down a problem into intermediate subgoals and subtasks and conducts a search for a network of methods and operators that accomplish the task, while a conceptual analyzer builds a representation of the natural language input by storing the CD structures generated by words in a working memory, and searching for ways to connect them together into larger CD structures (Riesbeck, 1975; Birnbaum & Selfridge, 1975).

Based on these observations, we argue for a view that goes beyond simply combining natural language generators with automated planners, or conjoining elements of actors' goals and plans with primitive decompositions of language. Instead we argue for a unified view in which primitive decomposition-based understanding and hierarchical planning are largely identical processes. This realization encourages work to use the structures and systems for hierarchical planning directly in decomposition of language for story understanding, for example, using the method specifications of the SHOP planner (Nau et al., 1999) in a conceptual analyzer such as ELI (Riesbeck, 1978) or Dypar (Dyer, 1982). It also encourages using structures for decomposition of natural language directly in hierarchical planning systems; a conceptual analyzer may have a conceptual lexicon entry for the word "walk" which could be used in the `travel-to` method for the SHOP planner. Obviously the CD conceptual primitives are not identical to the SHOP planner primitives, but this view encourages work on selecting domain-independent and reusable primitives, operators, and connectors for both kinds of systems. We illustrate this unified vision with an example in the following section.

4. Natural Language Primitive Decomposition: An Example

Figure 1 shows an example of a primitive decomposition for a simple sentence in Conceptual Dependency (Schank, 1972; Schank et al., 1975; Lytinen, 1992). In the figure, the sentence "Lisa kicked the ball to the fence" is decomposed into three connected primitive acts:

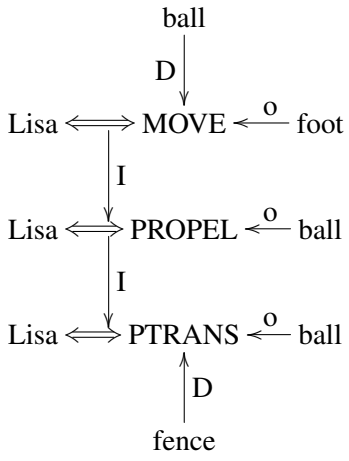

Figure 1. A CD decomposition of the sentence "Lisa kicked the ball to the fence." CD (Schank, 1972) decomposes the kicking act into a combination the primitive acts PTRANS, MOVE, and PROPEL. Double arrows indicate the actor performing the primitive act. Single arrows marked "o", "D", and "I" indicate the object, the directional case, and the instrument case, respectively.

- The PTRANS act represents only that Lisa in some way changed the location of the ball from one location to another; in this case from Lisa's location to being at the fence (indicated by the direction case link labeled "D"). For this act, Lisa is the actor and the ball is the object. How exactly Lisa performs the PTRANS is not specified by this primitive conceptualization alone.
- The PROPEL act represents only that Lisa in some way struck or applied a force to the ball. For this act, Lisa is the actor and the ball is the object. How exactly Lisa performs the PROPEL is not specified by this primitive conceptualization alone.
- The MOVE act represents only that Lisa moves a part of her body in a particular direction. With "Lisa's foot" in the "object" case, and "ball" in the direction case, it represents the primitive act of Lisa moving her foot in the direction of the ball.

The three acts are connected through the following links:

- The PTRANS act and the PROPEL act are connected via an "instrumental" link (indicated by the arrow labeled "I"), which indicates that the PROPEL act was performed as part of achieving the PTRANS and making it happen.
- The PROPEL act and the MOVE act are also connected via an "instrumental" link, indicating that the MOVE act was performed as part of achieving the PROPEL.

For story understanding, the decomposition of this sentence makes it simple to perform commonsense inferences based on a relatively small set of primitive acts instead of a potentially large

Conceptual Dependency Primitives	Pyhop Implementation

PTRANS: (Physical TRANSfer) To change the location of b
Pre: location(b) := x
Eff: location(b) := y
	`def ptrans(state,a,b,x,y):`
	`  if state.loc[b] == x:`
	`  state.loc[b] = y`
	`  return state`
	`  else: return False`

PROPEL: To apply a force to b
Pre: location(a) == location(b)
Eff: none.
	`def propel(state,a,b):`
	`  if state.loc[a] == state.loc[b]:`
	`  return state`
	`  else: return False`

MOVE: To move a body part
Pre: location(b) := x
Eff: location(b) := y
	`def move(state,a,b,x,y):`
	`  if state.loc[b] == x:`
	`  state.loc[b] = y`
	`  return state`
	`  else: return False`

Figure 2. Left: Descriptions, preconditions, and effects of three Conceptual Dependency primitives used to decompose the sentence "Lisa kicked the ball to the fence." Right: operators corresponding to the primitives implemented as functions in the Pyhop HTN planner.

collection of lexical items (for example, the PROPEL primitive act promotes an inference that the ball changed location). The decomposition also simplifies the understanding of paraphrases of identical stories. For example, the same conceptualization could be expressed as "Lisa struck the ball by moving her foot towards the ball". As long as conceptual analysis subsystems exist to decompose two paraphrases to a representation consisting of combinations of non-linguistic conceptual primitives, the paraphrases can be matched based on these structures, allowing the system to overcome many of the ambiguities of surface language forms.

5. SHOP as a CD Conceptual Analyzer

Pyhop[1] is a simple HTN planner with a planning algorithm that is nearly identical to that in SHOP (Nau et al., 2003). Pyhop is written in less than 150 lines of Python code, and it represents the planning state as a Python object containing variable bindings, and the HTN operators and methods as ordinary Python functions which refer to states explicitly. It is open-source software and available under the Apache License. We use the Pyhop framework to build a proof-of-concept HTN planner

1. http://bitbucket.org/dananau/pyhop

that performs primitive decomposition of natural language expressions into Conceptual Dependency conceptualization structures as described in Section 4. To accomplish the CD decomposition, we write three operators for `move`, `propel`, and `ptrans` (cf. Section 5.1), and we write a single method `kick` that decomposes lexical items into these operators (cf. Section 5.2).

5.1 CD Primitives as Planning Operators

The example in Figure 2 illustrates how the CD primitives can be implemented as operators within a planning system with appropriate preconditions and effects. For example, PTRANS in CD represents the act of changing the location of some object. The object can be the actor, meaning that the actor moved themselves to a different location, or the object can be someone or something other than the actor. PTRANS is nonspecific as to the exact way that the location change is effected. We implement PTRANS as the operator `ptrans(state,a,b,x,y)` with the arguments `a` and `b` representing the actor and object, and the arguments `x` and `y` representing the initial and final location of the object. The precondition for PTRANS is that the location of the object is `x`, and the effect is that the location of the object is `y`.

PROPEL represents the act of applying force to an object as well as the act of striking or colliding with an object. A PROPEL does not necessarily mean that the object moves or accelerates, but it does as the instrument of a PTRANS in this case. We implement PROPEL as the operator `propel(state,a,b)` with the arguments `a` and `b` representing the actor and object. The precondition for PROPEL is that the locations of the actor and the object are the same, and it has no effects on its own.

MOVE represents the actor moving a part of their body in the direction of another object, but not necessarily changing location of their entire body. We implement MOVE as the operator `move(state,a,b,x,y)` with the arguments `a` and `b` representing the actor and the object (the body part that MOVEs) and the arguments `x` and `y` representing the initial and final location of the body part. The precondition for MOVE is that the location of the body part is `x`, and the effect is that the location of the body part is `y`.

5.2 Lexical Items as High-Level Tasks

Figure 3 shows how we implement words in natural language as high-level methods of the hierarchical planning process, establish the initial state of the system for the planner, and execute the planning process with words in the sentence as the high-level method and arguments to that method. The "kick" method is implemented as `kick(state,a,b,x,y)` with four arguments: `a` and `b` representing the actor and the object that gets kicked, and `x` and `y` representing the initial and final location of the object. The implementation of kick has the actor, `a`, `move` their foot from an unknown location toward the object, a `propel` of the object, and a `ptrans` by the actor, changing the location of the object from `x` to `y`.

The planner method, `pyhop.pyhop`, is called with the state object and the task. The code creates a state object for initial state of the world, `state1`, and initializes it with a member named `loc` representing the location of each object as hashtable pairs. The initial state has both Lisa and the ball at location `Lisa`, while the location of Lisa's foot is unspecified. Words from the sentence

```
def kick(state,a,b,x,y):
    return [('move',a,'foot','?',b),
            ('propel',a,b),
            ('ptrans',a,b,x,y)]

state1 = pyhop.State('state1')
state1.loc = {'Lisa':'Lisa','foot':'?','ball':'Lisa'}

pyhop.pyhop(state1,[('kick','Lisa','ball','Lisa','fence')])
```

```
** pyhop, verbose=1: **
   state = state1
   tasks = [('kick', 'Lisa', 'ball', 'Lisa', 'fence')]

** result = [('move','Lisa','foot','?','ball'),
           ('propel','Lisa', 'ball'),
           ('ptrans','Lisa','ball','Lisa','fence')]
```

Figure 3. Top: Pyhop implementation of the "kick" method as a sequence of operators: `move`, `propel`, and `ptrans`. Code to establish the initial state, `state1` consisting of initial locations of Lisa, Lisa's foot (initial location unknown), and the ball (at or near Lisa), and the call to the Pyhop planner with the task being a kick "high-level" act. Bottom: output from the Pyhop planner with the resulting hierarchical task network plan that decomposes the act from high levels of abstraction to the primitive operators representing CD primitives, `move`, `propel`, and `ptrans`.

are fed into the planning process as arguments to the planner that include "kick" as the high level task, and its arguments extracted from the sentence: "Lisa" as the actor, "ball" as the object being kicked, "Lisa" as the initial location of the object, and "fence" as the final location of the object after the kicking task is completed.

Executing the planner results in a conceptual analysis of the story that consists of the primitive operators `move`, `propel`, and `ptrans`, with the appropiate arguments for each based on the high-level task specification.

5.3 Discussion

The implementation of the kick method has three operators which are instances of CD primitive acts that are meant to correspond to the CD conceptualization of the kick as a PTRANS with a PROPEL

as its instrument, and a MOVE as an instrument of the PROPEL. However, the kick method has the operators as a *sequence* of primitive acts, in the order MOVE, PROPEL, and PTRANS; this was implemented so that the preconditions and effects of the primitive acts meet together properly, with the effects of the PTRANS resulting in the location of the ball being at the fence.

Unfortunately this does not preserve the hierarchy of instrumentation relationships, and simply presents the primitive acts as a sequence. The purpose of the instrumentation relationship between the acts is to answer "how" questions relating the various acts. A classic CD story understander (such as that in Dyer, 1982, or Lehnert, 1978) would answer a question like "How did Lisa get the ball to the fence?" by corresponding "get the ball to the fence" to the PTRANS act, and following the instrumental link from the PTRANS to the PROPEL instrumented by MOVE, and could answer: "Lisa moved her foot towards the ball and struck it."

This raises the issue of how instrumental links and other causal links between primitive conceptualizations should be represented in the HTN conceptual analysis system. One might have the planner's hierarchy represent CD instrumentation links. In our example, this could be achieved by making the PTRANS be a separate method whose implementation is just the PROPEL, and making the PROPEL a separate method and making its implementation be the MOVE act. But in our case this has the unintended consequence of making PTRANS overspecialized, so that all PTRANS acts are implemented through kicking. If we continue our example from Section 2.4, we would also like to implement a method for the word "punch" alongside our method for "kick". "Punch" should be implemented with a PTRANS, but with a PROPEL that is instrumented by a MOVE of the actor's fist instead of their foot.

For hierarchical planning, methods usually may have multiple implementations, and it is the task of the planner to choose the "best" implementation through optimization. One could add a second implementation for PTRANS to represent "punch". But then the planner will not differentiate between the two implementations when analyzing a text having "punch" versus a text having "kick"—unless, of course, the state of the parse, including which word was being decomposed, is part of the state space of the planner. In this case the PTRANS method could select between "punch" and "kick" implementations by consulting this parse state information.

There has long been recognition that, in language understanding systems, the purpose of representing plans and the links between actions in plans is explanatory, while, in most planning systems, the purpose is generating the plan (Schank & Abelson, 1977). Generally, more investigation is needed on the best methods for unification of hierarchical planning, in which the only links between acts in plans are through the hierarchy, and conceptual analysis, which has a variety of link types that form connections between primitive acts.

6. Conclusion

This position paper explores the connections between hierarchical planning and natural language understanding and argues for a unified view of conceptual primitive decomposition of natural language and hierarchical planning. This view promotes novel studies of primitive decomposition systems that can cross between the two domains, and the reuse of methods, operators, and knowledge structures from each domain. It establishes another unique route towards richer integration between

planning systems and natural language systems, which could provide natural language understanding and natural language generation capability from within automating planners to better understand planning problem statements and explain the plans they generate. At the same time, work according to this vision could provide deeper and more direct connections to powerful planning systems for in-depth natural language understanding and story generation systems.

We also demonstrated a prototype of a hierarchical task network planning system that implements a conceptual primitive analyzer of natural language. Although HTN planners are similar to conceptual analyzers in the way that they attempt to decompose tasks or decompose words into combinations of primitive elements, we did encounter a number of practical issues to address in immediate future work. While HTN systems do decompose in multiple levels of abstraction from high-level methods to primitive operators, they may not preserve relationships between acts to be used for explanation, as shown in the example of the "instrument" case links between primitive conceptualizations used to answer "why" questions about the acts. In future work it may be possible that the differentiation of "causal" connectives between primitive acts in CD could be reconciled with or integrated with the precondition and effect relationships between acts as specified in the planner. As we mentioned earlier, a more recent variant of hierarchical planning called Hierarchical Goal Networks (HGNs) blends the best of sequential and hierarchical planning. We plan to explore the benefit of representing portions of Conceptual Dependency hierarchies as HGNs.

Also, to make an HTN planner more like a conceptual analyzer, which takes actual natural language as input, it is clear that some form of preprocessing is needed to properly invoke the planner with the correct arguments of the high-level task ("kick" in our example). While a parser or part-of-speech tagger may seem like a good choice initially to perform this function, to be more true to the conceptual analysis tradition encourages research in approaches that integrate parsing more strongly with the planner decomposition.

Specifying the domain for such systems is a tedious and error-prone activity as it relies on hand-coded effort. Another direction we plan to explore is to automatically learn the language hierarchies from traces of actions (e.g., using techniques such as that of Hogg et al., 2009), which would greatly reduce the burden of domain designers. Automated approaches could leverage the work of existing ontologies to bootstrap an initial set of language hierarchies. From this, hierarchies could be refined using techniques such as those from the Never Ending Language Learner (Mitchell et al., 2018). Finally, these system integration efforts may allow for planning systems to use knowledge structures from language understanders and for language understanders to use knowledge from planning systems; further work on these knowledge engineering issues is needed.

While we have shown that a planner can be used to perform conceptual analysis on language, we envision using planning for commonsense reasoning and inference about narratives. One of the motivations of integrating planning systems in natural language or narrative understanding is the ability to reason that certain acts occurred in the story (or can be predicted to occur in the story) even when they were not stated explicitly, based on the reasoned goals and plans of story actors. Our prototype does not yet do that kind of planning, but this may be a product of future integration. If the narrative understanding process is "reversed" into natural language generation, this work may enable generation of better explanations of plans generated by HTN systems. Finally, there are obviously further deep connections between primitive decomposition, natural language

understanding, and planning through Schank-Abelson scripts (Schank & Abelson, 1977) that have yet to be explored.

Acknowledgements

The authors thank the Naval Research Laboratory and the Office of Naval Research Summer Faculty Research Program for supporting this work. We also thank both Sandra Grandic and the anonymous reviewers for their thoughtful comments which led to improvements of this paper.

References

Alford, R., Shivashankar, V., Roberts, M., Frank, J., & Aha, D. W. (2016). Hierarchical planning: Relating task and goal decomposition with task sharing. *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence* (pp. 3022–3028). New York: AAAI Press.

Allen, J. (1995). *Natural language understanding*. London: Pearson.

Baker, C. F., Fillmore, C. J., & Lowe, J. B. (1998). The Berkeley FrameNet project. *Proceedings of the Thirty-Sixth Annual Meeting of the Association for Computational Linguistics and Seventeenth International Conference on Computational Linguistics* (pp. 86–90). Montreal: ACL.

Banarescu, L., et al. (2013). Abstract meaning representation for sembanking. *Proceedings of the Seventh Linguistic Annotation Workshop and Interoperability with Discourse* (pp. 178–186). Sofia, Bulgaria: ACL.

Berant, J., Chou, A., Frostig, R., & Liang, P. (2013). Semantic parsing on freebase from question-answer pairs. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (pp. 1533–1544).

Birnbaum, L., & Selfridge, M. (1975). Conceptual analysis of natural language. In R. C. Schank & C. K. Riesbeck (Eds.), *Inside computer understanding: Five programs plus miniatures*, 318–353. Hillsdale, NJ: Lawrence Erlbaum.

Cambria, E., Poria, S., Bajpai, R., & Schuller, B. W. (2016). SenticNet 4: a semantic resource for sentiment analysis based on conceptual primitives. *Proceedings of the Twenty-Sixth International Conference on Computational Linguistics* (pp. 2666–2677). Osaka, Japan: The COLING 2016 Organizing Committee.

Cambria, E., Poria, S., Hazarika, D., & Kwok, K. (2018). SenticNet 5: Discovering conceptual primitives for sentiment analysis by means of context embeddings. *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. New Orleans, LA: AAAI Press.

Diakidoy, I.-A., Kakas, A., Michael, L., & Miller, R. (2015). STAR: a system of argumentation for story comprehension and beyond. *Proceedings of the 12th International Symposium on Logical Formalizations of Commonsense Reasoning* (pp. 64–70). Stanford, CA: AAAI Press.

Dyer, M. G. (1982). *In-depth understanding: a computer model of integrated processing for narrative comprehension*. Cambridge, MA: MIT Press.

Erol, K., Hendler, J. A., & Nau, D. S. (1994). UMCP: a sound and complete procedure for hierarchical task-network planning. *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems* (pp. 249–254). Chicago, IL: AAAI Press.

Geib, C. (2015). Lexicalized reasoning. *Proceedings of the Third Annual Conference on Advances in Cognitive Systems*. Atlanta, GA: The Cognitive Systems Foundation.

Haslum, P. (2012). Narrative planning: Compilations to classical planning. *Journal of Artificial Intelligence Research*, *44*, 383–395.

Hogg, C., Kuter, U., & Muñoz-Avila, H. (2009). Learning hierarchical task networks for nondeterministic planning domains. *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence* (pp. 1708–1714). Pasadena, CA: AAAI Press.

Jackendoff, R. S. (1983). *Semantics and cognition*. Cambridge, MA: MIT Press.

Lehnert, W. G. (1978). *The process of question answering: A computer simulation of cognition*. Hillsdale, NJ: Lawrence Erlbaum.

Lytinen, S. L. (1992). Conceptual dependency and its descendants. *Computers and Mathematics with Applications*, *23*, 51–73.

Macbeth, J. C. (2017). Conceptual primitive decomposition for knowledge sharing via natural language. *Proceedings of The Joint Ontology Workshops, Episode 3: The Tyrolean Autumn of Ontology*. Bolzano-Bozen, Italy: The International Association for Ontology and its Applications.

Macbeth, J. C., & Barionnette, M. (2016). The coherence of conceptual primitives. *Proceedings of the Fourth Annual Conference on Advances in Cognitive Systems*. Evanston, Illinois: The Cognitive Systems Foundation.

Macbeth, J. C., & Grandic, S. (2017). Crowdsourcing a parallel corpus for conceptual analysis of natural language. *Proceedings of The Fifth AAAI Conference on Human Computation and Crowdsourcing* (pp. 128–136). Quebec City, Quebec, Canada: AAAI Press.

Macbeth, J. C., Gromann, D., & Hedblom, M. M. (2017). Image schemas and conceptual dependency primitives: a comparison. *Proceedings of the Joint Ontology Workshops, Episode 3*. Bolzano-Bozen, Italy: International Association for Ontology and its Applications.

Minsky, M. (1988). *Society of mind*. Simon & Schuster.

Mitchell, T., et al. (2018). Never-ending learning. *Communications of the ACM*, *61*, 103–115.

Nau, D., Cao, Y., Lotem, A., & Muñoz-Avila, H. (1999). SHOP: Simple hierarchical ordered planner. *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence* (pp. 968–973). Stockholm: Morgan Kaufmann.

Nau, D. S., Au, T.-C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., & Yaman, F. (2003). SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research*, *20*, 379–404.

Norvig, P. (1983). Frame activated inferences in a story understanding program. *Proceedings of the Eighth International Joint Conference on Artificial Intelligence* (pp. 624–626). Karlsruhe, Germany: AAAI Press.

Petrick, R., & Foster, M. E. (2013). Planning for social interaction in a robot bartender domain. *Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling*. Rome: AAAI Press.

Porteous, J., & Cavazza, M. (2009). Controlling narrative generation with planning trajectories: The role of constraints. *Proceedings of the Second International Conference on Interactive Digital Storytelling* (pp. 234–245). Guimaraes, Portugal: Springer.

Porteous, J., Charles, F., & Cavazza, M. (2013). NetworkING: Using character relationships for interactive narrative generation. *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems* (pp. 595–602). Saint Paul, MN: International Foundation for Autonomous Agents and Multiagent Systems.

Riedl, M. O., & Young, R. M. (2010). Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research*, *39*, 217–268.

Riesbeck, C. K. (1975). Conceptual analysis. In R. C. Schank (Ed.), *Conceptual information processing*, 83–156. New York: Elsevier.

Riesbeck, C. K. (1978). An expectation-driven production system for natural language understanding. In D. A. Waterman & F. Hayes-Roth (Eds.), *Pattern-directed inference systems*, 399–413. New York: Elsevier.

Rosenthal, S., Selvaraj, S. P., & Veloso, M. M. (2016). Verbalization: Narration of autonomous robot experience. *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence* (pp. 862–868). New York: AAAI Press.

Schank, R. C. (1972). Conceptual dependency: A theory of natural language understanding. *Cognitive Psychology*, *3*, 552–631.

Schank, R. C., & Abelson, R. P. (1977). *Scripts, plans, goals and understanding: an inquiry into human knowledge structures*. Hillsdale, NJ: Lawrence Erlbaum.

Schank, R. C., Goldman, N. M., Rieger III, C. J., & Riesbeck, C. K. (1975). Inference and paraphrase by computer. *Journal of the ACM*, *22*, 309–328.

Shi, L., & Mihalcea, R. (2005). Putting pieces together: Combining FrameNet, VerbNet and WordNet for robust semantic parsing. *Proceedings of the Sixth International Conference on Intelligent Text Processing and Computational Linguistics* (pp. 100–111). Mexico City: Springer.

Shivashankar, V., Alford, R., Kuter, U., & Nau, D. (2013). The GoDeL planning system: A more perfect union of domain-independent and hierarchical planning. *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence* (pp. 2380–2386). Beijing: AAAI Press.

Shivashankar, V., Kuter, U., Nau, D., & Alford, R. (2012). A hierarchical goal-based formalism and algorithm for single-agent planning. *Proceedings of the Eleventh International Conference on Autonomous Agents and MultiAgent Systems* (pp. 981–988). Valencia, Spain: International Foundation for Autonomous Agents and Multiagent Systems.

Speer, R., & Havasi, C. (2013). ConceptNet 5: A large semantic network for relational knowledge. In I. Gurevych & J. Kim (Eds.), *The people's Web meets NLP*, 161–176. New York: Springer.

Wierzbicka, A. (1996). *Semantics: Primes and universals*. New York: Oxford University Press.

Wilensky, R. (1983). *Planning and understanding: A computational approach to human reasoning*. Reading, MA: Addison-Wesley.

Wilks, Y., & Fass, D. (1992). The preference semantics family. *Computers and Mathematics with Applications*, *23*, 205–221.

Winograd, T. (1972). *Understanding natural language*. New York: Academic Press.

Winston, P. H. (2011). The strong story hypothesis and the directed perception hypothesis. *Papers from the 2011 AAAI Fall Symposium on Advances in Cognitive Systems* (pp. 345–352). Arlington, VA: AAAI Press.

Winston, P. H. (2014). *The Genesis story understanding and story telling system: A 21st century step toward artificial intelligence*. Technical report, Center for Brains, Minds and Machines, Massachusetts Institute of Technology, Cambridge, MA.