Knowledge-Based Systems 000 (2017) 1–12

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Generating machine-executable plans from end-user's natural-language instructions

Rui Liu, Xiaoli Zhang*

Department of Mechanical Engineering, Colorado School of Mines, Golden, CO, 80401, USA

ARTICLE INFO

Article history:
Received 15 March 2017
Revised 2 September 2017
Accepted 20 October 2017
Available online xxx

Keywords:
Semantic analysis
Machine-executable plan
Natural language instruction
Advanced manufacturing machine

ABSTRACT

It is critical for advanced manufacturing machines to autonomously execute a task by following an end-user's natural language (NL) instructions. However, NL instructions are usually ambiguous and abstract so that the machines may misunderstand and incorrectly execute the task. To address this NL-based human-machine communication problem and enable the machines to appropriately execute tasks by following the end-user's NL instructions, we developed a Machine-Executable-Plan-Generation (exePlan) method. The exePlan method conducts task-centered semantic analysis to extract task-related information from ambiguous NL instructions. In addition, the method specifies machine execution parameters to generate a machine-executable plan by interpreting abstract NL instructions. To evaluate the exePlan method, an industrial robot Baxter was instructed by NL to perform three types of industrial tasks {"drill a hole", "clean a spot", "install a screw"}. The experiment results proved that the exePlan method was effective in generating machine-executable plans from the end-user's NL instructions. Such a method has the promise to endow a machine with the ability of NL-instructed task execution.

1. Introduction

Human-machine collaborative manufacturing combines human intelligence on high-level task planning and the robot physical capability (e.g., precision and speed) on low-level task execution [1]. Toward this direction, intuitive and natural communication between the human and the machine has been an active research area in the last decade with the goal to enable seamless human-machine cooperation [2,3]. Natural-Language-instructed human-machine interaction is expected to enable an advanced manufacturing machine, such as a Computer Numerical Control machine or an industrial robot, to autonomously perform tasks such as rough/fine finishing [4,5], assembly [2,6] and packaging [7,8] according to the end-user's NL instructions, which are given based on the user's judgement of the task progress and environmental situations. Compared with other input methods, including human hand force [9,10], hand gesture [11,12], and body motions [13–16], the NL instruction method has two main advantages. First, NL instruction provides a natural, human-like, face-to-face communication manner. Non-expert users without prior programming training could command a machine to perform their desired tasks [17,18]. Second, the inherent linguistic structure of NL, as a predefined information encoder, provides a standard, informative data source to generate structured machine language [19,20]. In contrast, the aforementioned existing methods require extra translations among discrepant data patterns. These two advantages make NL a superior means for the end-user to naturally and efficiently communicate with manufacturing machines.

Currently, typical industrial applications involving NL include NL-based control in which the working statuses such as "on/off" and "quickly/slowly" are selected orally to control a machine in navigation [3,21], NL-based task execution in which the task operation methods such as "goTo + Location; then drop + object" is described orally to help a machine with object finding/placing [22,23], and NL-based execution personalization in which human's preferences and moods in oral dialogs were considered to adjust a machine's execution manners [24,25].

However, there is still a long way to apply NL-instructed machines in practical manufacturing applications. First, NL is variable and ambiguous. NL is usually polysemous, homophonic and expression-manner diverse so that the same meaning could be expressed in various ways, and different meanings could be expressed in similar ways [8,26]. For example, "drill a hole" could be expressed as "bore one hole", "drilling one bore", "create an unthreaded hole", and so on [27]. In addition, humans usually use referring, outlining, and omitting in NL instructions [22,28]. For example, in an instruction "at the center point", information such as "which object in which place has the center point" cannot be known merely from a word 'the' [27]. It is challenging to ex-

* Corresponding author.
E-mail addresses: rliu@mines.edu (R. Liu), xlzhang@mines.edu (X. Zhang).

https://doi.org/10.1016/j.knosys.2017.10.023
0950-7051/

tract task-related information such as task goals and detailed execution procedures from NL instructions. Second, human instruction is abstract [23,29]. Even when a complete execution procedure for a task is instructed, the generated plan is still non-executable for a machine. For example, the abstract instructions 'clean the surface' are still machine-non-executable for that the execution-related specific knowledge such as "tool: brush; action: moveDown → sweep → moveUp; ..." is missing [27]. In addition to specific-knowledge missing, a reasonable and flexible knowledge structure, which is implicitly embedded in NL descriptions to guide correct task execution, is difficult to extract [30–32]. By obeying the human instructions, one task could be flexibly executed by several methods, which were formulated according to an individual's cognitive logics [33,34]. However, usually these cognitive logics in NL instructions are difficult to understand as to a machine, for that literal information directly extracted from NL instructions is insufficient to explain the logics [2,35]. Taking the task "deliver a drink" as an example, the potential methods could either be "fetch a cup + fill the cup with water + place it on table" or "place a cup on the table + fill cup with water". The logics such as {*CupAvailability(yes)* ^ *WaterAvailability(yes)* → *CupBeingFilledFeasibility(yes)*} behind the task executions have not been described explicitly in NL instruction while these logics are important in deciding what kind of procedures are feasible and reasonable in execution and in assessing whether a task could be executable or not. It is challenging for a machine to perform a task without knowing the task-related logics.

To address these problems and enable NL-instructed manufacturing in practical industrial tasks, we developed a machine-executable-plan-generation (exePlan) method to "translate" the ambiguous and abstract NL instructions into machine-executable plans. In this paper, we mainly have two contributions, shown as follows.

- A task-centered semantic analysis method is developed for processing ambiguous NL instructions into task-related information including task goal, sub-goals, and execution logic relations. Instead of using basic linguistic features such as keywords/Part-of-Speech(PoS), the task-related semantic features, such as actions/tools/execution logics were considered to extract the task-related information from ambiguous NL instructions.
- A machine-execution-specification method is developed for interpreting abstract human instructions into machine-executable plans. With this method, each abstract sub-goal in the NL instruction is firstly specified into an executable sub-goal by adding the machine-execution specification (MES) parameters such as location, action, and human requirements. Then a machine-executable plan is specified by exploring the weighted logic relations among the task-related execution procedures. Different from the first-order logic in which all the logic relations are inviolable and plans using first-order logic have binary executability {executable, non-executable}, weighted logic relations could be violable with a corresponding weighted decrease of plan executability and the plans using weighted logics have a range of acceptable executability. A plan is flexibly made by organizing reasonable logic procedures with the executability greater than a threshold value.

2. Related work

To disambiguate NL instructions in task execution, special grammars were designed to identify the task-related entities based on specific keyword involvements and their PoS tags. For example, in the sentence "bring the can in the trash bin" the task goal "in the trash bin" was extracted based on the keywords "bring, can" and their corresponding PoS tags "VB, NN' [8]. Ontology relations among the interested entities were used for mutual disambiguation. For example, to describe a cup, the description was likely to be "container with handle attached". "Attached" was the constraint relation between the object "container" and object part "handle" [36,37]. When an entity was ambiguously mentioned, the ambiguous entity could be explained by mutually co-referring. Take sentences "Go to the second crate on the right. Pick it up" for an example, with co-reference resolution the uncertain expression "it" was identified as "the second crate on the right" [22,38]. When the NL descriptions such as "pick up the pallet" were too ambiguous for a robot, a query such as "which pallet?" was launched to ask the human for disambiguation [2,39]. By exploring the features such as perceivable properties "cylindrical" and "round", the ambiguous descriptions "cylindrical container with a round handle attached on one side" for the object "container" was understood [36]. By exploring the spatial relations "behind" in NL descriptions "Navigate to the building behind the pole", named entities "building, pole" were identified in the real world [40,41]. To disambiguate the NL instructions, these methods explored context evidences for a single entity. Evidences include semantic relations, human explanations, and spatial constrains. However, these methods only focused on using one single type of evidences such as basic linguistic feature keywords/PoS or semantic features co-referring and perceivable properties, without combining the multiple types of features together to perform a comprehensive semantic analysis. In addition, these methods aimed to identify an entity such as "can" or "trash bin" separately without considering entity correlations such as "can—trash bin", which are informative in instruction disambiguation. The above mentioned features are important for semantic analysis, however have not been well investigated.

To interpret abstract expressions in NL instructions, motion grammars were first designed for establishing the word-action correlations such as word "grasp" — action "Grasp" [3,21,23]. Real-world preconditions such as "stay in the kitchen" were defined for triggering specific types of executions such as "visiting the kitchen" [17]. The NL descriptions were marked by landmark objects such as "staircase, box" in the real world to enable the execution of tasks such as "reach in a spot" [42,43]. With these methods, abstract NL descriptions are interpreted into executable commands to some extent. However these methods do not make the NL command truly machine-executable for that the critical execution parameters, including the tool usage, real-world precondition, action sequence, and human requirements, are still missing or insufficient for supporting a robot's executions in practical situations.

To interpret implicit cognitive logics embedded in NL instructions, probabilistic graphical models were designed to explore the knowledge importance with the consideration of its probability distributions for plan making. For example, in the NL descriptions "go to the second crate on the right. Pick it up.", the procedures could be modeled as {goTo create ($p = .50$), PickUp crate ($p = .50$)} [22,43]. In the study of human-speech-instructed indoor navigation [46,47], a semantic topological model was developed to explore the internal logic correlations of sub-steps in a reasonable task-execution plan. For example, in the path "first go to the hallway; the cafeteria is down the hallway" the hallway could be replaced by "hall, corridor, walkway" and the "cafeteria" could be replaced by "dining hall" according to the semantic topology. The procedures with any combinations of the elements in the topology structure were considered as reasonable task execution plans. However, these plans are not truly executable for a machine. Probabilistic graphical models merely describe the importance of the sub-steps in a plan, ignoring their internal logics without which a plan is non-executable in the real world. On the other hand, topological models describe the logic relations among procedures; however, the logic constrains are hard without discriminative descriptions of the involved logic relations. If one hard logic relation

Please cite this article as: R. Liu, X. Zhang, Generating machine-executable plans from end-user's natural-language instructions, Knowledge-Based Systems (2017), https://doi.org/10.1016/j.knosys.2017.10.023