

ABC: Automatic Bottom-up Construction of Configuration Knowledge Base for Multi-Vendor Networks

Wenlong Ding* Libin Liu† Li Chen† Hong Xu*

*The Chinese University of Hong Kong †Zhongguancun Laboratory

Abstract—The Configuration Knowledge Base (CKB) facilitates network configuration management in multi-vendor environments by storing configuration snippet templates and parameter settings for various vendors and intents, simplifying the deployment of the same configuration intent across different vendor devices. Current CKB construction methods follow a top-down manner, which requires handcrafted parameter labeling and snippet template creation for every new vendor or intent, resulting in significant expert effort. We seek to reduce this heavy human effort by leveraging our insights that different vendors share similar underlying intents and corresponding parameter types despite their different configuration languages, motivating bottom-up CKB construction manners which automatically create and align snippet templates of the same intent across all vendors by analyzing already existing device manuals and network configuration files. More specifically, we introduce **ABC**, an active-learning-based tool that utilizes NLP models to produce snippet templates with manuals, extract labeled parameters in existing configuration files, and align those templates with common configuration models by encoding and comparing manuals of different vendors. We believe that creating such a tool would be a significant step towards achieving highly *automated network configuration*, which would also serve as a solid foundation for future endeavors such as intent-based network configuration, network configuration synthesis, and network device assimilation.

I. INTRODUCTION

Nowadays, large enterprises usually build their network infrastructures with devices from multiple vendors [6, 9, 13, 14]. This multi-vendor deployment provides notable benefits, including cost savings through the availability of comparable product options from diverse vendors, integration of unique functionalities offered by different vendors’ devices, and risk mitigation in case a vendor discontinues a specific type of device. As enterprises grow and evolve, the need for different network features at various stages drives the continuous introduction of new vendors, emphasizing the importance of effective configuration management and configuration settings sharing across vendors.

Network operations (NetOps) teams from various enterprises have devoted considerable effort to establishing centralized multi-vendor network management platforms [4, 5, 16]. These platforms facilitate unified tools for operation, administration, and management tasks in the network. At their core, these platforms incorporate a logically centralized repository known as Configuration Knowledge Base (CKB), which serves as a comprehensive storehouse maintaining configuration information for devices across multiple vendors. Specifically, the CKB encompasses unified network parameters or settings information, as well as vendor-specific configuration templates

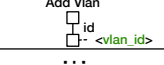

Intent	Config Snippet Templates		Unified Config Model	
	Cisco	...		Huawei
Add VLAN	vlan <vlan_id>	...	vlan branch <vlan_id>	Add Vlan 
...
Add ACL	access-list <ACL number> {permit deny} <protocol> source <ip> <mask> destination <ip> <mask>	...	acl number <ACL number> rule <rule number> {permit deny} <protocol> source <ip> <mask> destination <ip> <mask>	ACL Config 

Fig. 1: Example of the configuration knowledge base (CKB). CKB contains configuration snippet templates for different vendors and intents with a unified configuration model to store parameter settings of each intent across vendors.

that can collaborate with the unified information to generate specific configurations for any vendor or intent, as the example shown in Fig. 1. Essentially, the CKB normalizes, organizes, and stores configuration details for all devices from different vendors within a given network autonomous domain.

Existing works propose unified configuration models to store parameter settings and relationships in organized structures, such as FBNet in Robotron [16], configuration tree in OpenConfig [5], and vendor parsing model in Nassim [8], which can facilitate convenient configuration settings sharing across different vendor devices. These works adopt a top-down approach to construct CKB for diverse configuration languages in different vendors. In this process, experts handcraft vendor-specific configuration templates without parameter values based on target high-level intents, identify and store parameter settings in the unified model, and finally fill configuration templates with these unified settings for low-level snippets of a specified intent and vendor. This top-down approach requires expert knowledge of various vendors’ configuration languages and continuous human effort in handwriting templates for every new vendor and intent throughout the network’s lifecycle.

We aim to automate CKB construction by reducing human effort in traditional top-down approaches. We believe the main challenge lies in significant differences in vendor-specific configuration languages, which cannot be fundamentally eliminated because it stems from vendors intentionally to discourage vendor-switching of customers [3]. However, we recognize that different vendors share similar high-level intents and parameter types of the same intent despite language differences. Based on this insight, we propose **ABC**, a novel **Automatic Bottom-up Construction** approach for CKB with NLP models. It first learns parameter extraction rules to generate configuration templates from low-level example snippets of different intents in device manuals, without focusing on

Intent	Cisco	Huawei	Juniper
<i>check vlan</i>	show <u>vlan</u> [vlanid]	display <u>vlan</u> [vlanid]	show <u>vlan-id/vlans</u> [vlanid]/[vlanname]
<i>add/delete vlan</i>	<u>vlan</u> [vlanid]/no <u>vlan</u> [vlanid]	<u>vlan branch</u> [vlanid]/undo <u>vlan branch</u> [vlanid]	set <u>vlan-id</u> [vlanid]/delete <u>vlan-id</u> [vlanid]
<i>configure spanning tree root bridge</i>	spanning tree <u>vlan</u> [vlanid] root primary	stp instance [vlanid] root primary	spanning-tree <u>vlan-id</u> [vlanid] root primary

TABLE I: Config syntax comparisons between three different vendors: Cisco, Huawei, and Juniper.

Intent	Cisco IOS	Cisco IOS XR
<i>Assign IP address and subnet to interface</i>	ip address [ip/prefix]	ipv4 address [ip] [submask]
<i>Receive LLDP packets on an interface</i>	lldp receive	lldp admin-status rx
<i>Set the interface as an Ethernet trunk</i>	switchport mode trunk	port link-type trunk

TABLE II: Configuration syntax comparisons between different OSs within the same vendor (Cisco): Cisco IOS & Cisco IOS XR.

specific syntax rules of configuration languages. It then aligns templates of the same intents across vendors with their intent descriptions in manuals and finally fills them with extracted parameters from the target configuration files, enabling consistent deployment of high-level target intents across the network.

ABC comprises three modules: parameter parsing (PP), template alignment by intent (ALIGN), and snippet extraction. PP uses Named entity recognition (NER) models to extract labeled parameters from example snippets and generates templates with parameter placeholders. ALIGN leverages BERT-based models [17] to align snippet templates of the same intents to a unified Common Configuration Tree (CCT) in OpenConfig [5], based on the similarity of encoding results of various templates and their intent descriptions. Snippet extraction splits target configuration files into intent-specific snippets, then use PP to extract labeled parameters and store them in CCT. ABC has two workflows: Intent-Template-CCT (ITC) mapping for producing aligned templates and parameter extraction rules, and Target Configuration Storing for extracting and storing labeled parameters of target intents.

Automated network configuration is a significant topic for NetOps teams, aiming to generate low-level network configuration across different vendors based on high-level human intents, reducing human effort in learning different configuration languages and determining parameter settings for specific intents. ABC is a notable contribution in this field, automatically constructing an intent-based CKB using learning methods, reducing expert knowledge and handcrafted works compared to traditional top-down approaches. Its success will be marked as a milestone and inspire further research in this topic, such as *Intent-based Network Configuration* and *Network Configuration Synthesis*, which will analyze high-level intents directly for parameter settings and synthesize configuration files or templates across multiple intents, respectively. Further insights into potential vision works can be found in §IV.

II. BACKGROUND AND MOTIVATION

In this section, we first introduce existing top-down CKB construction methods, then motivate our bottom-up approach, and finally highlight the background and uniqueness of the knowledge extraction task in ABC.

A. Limitations of Top-down CKB Construction

CKB simplifies multi-vendor network management by storing configuration templates by intents across vendors with parameter settings stored in a unified manner, as shown in

Fig. 1. Existing works propose unified configuration models to store parameter settings and their relationships in an organized structure. For example, Facebook’s Robotron [16] uses FBNet with tables to store parameters and directed links to present the relationship between parameters, while Google’s OpenConfig [5] adopts a tree structure called common configuration tree (CCT) to store settings and represent hierarchical relationships in most configuration languages. Apstra [1] introduces a graph-based representation for configuration storage, and Nassim [8] builds a tree structure similar to CCT with a unified snippets parsing model which has a set of rules allowing NetOps teams to handcraft program functions to map vendor-specific configurations to that tree structure.

However, these works take significant human effort to construct CKB in a top-down manner: NetOps teams must prepare vendor-specific templates (or parsing functions in Nassim [8]) for high-level intents, identify parameter settings or extract them from target configuration files which will be stored in the unified configuration model afterwards, and finally generate low-level configuration snippets across different vendors with templates and parameters. This demands expert knowledge of various configuration languages and human effort to provide snippet templates for new vendors or intents throughout the network’s lifecycle. This effort involved is substantial in the production network. For example, our experience with one of the largest NetOps teams spent nearly two years developing vendor-specific templates for $O(10^5)$ routers for six vendors based on CCT in OpenConfig [5]. Therefore, it is crucial to develop an automated system for constructing the CKB.

B. Motivation for Bottom-up CKB Construction

We believe the key difficulty in top-down CKB construction arises from intentionally different configuration languages and patented syntax [3], preventing a clear configuration templates mapping between vendors and requiring the creation of repeated templates for the same intent across vendors. We detail the configuration language differences in two aspects:

- **Syntax diversity across vendors.** Different vendors use varied wordings and organizations for the same configuration intent [3, 8], as demonstrated in Table I for Cisco, Huawei, and Juniper. This necessitates the creation of separate configuration templates for each vendor, even for simple tasks. There are also unique statements in vendor-specific configuration languages, such as the `disable-client-reflect` statement in Cisco [3], which does not exist in other vendor languages.
- **Syntax drift for the same vendor.** Even within the same vendor, configuration snippets for the same intent may differ across different operating system (OS) versions. For instance, a comparison between Cisco IOS and IOS XR in Table II reveals differences in wordings and statement

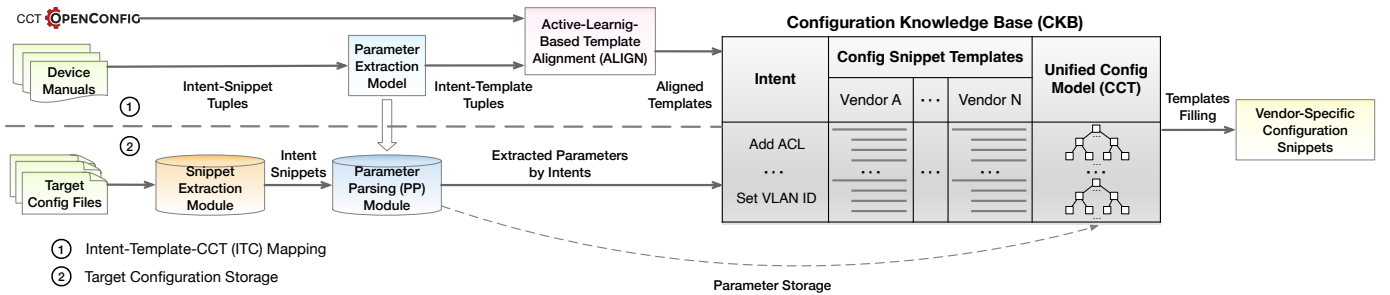


Fig. 2: ABC operational overview. It has two workflows: ITC mapping creates config templates of various intents and vendors by learning from manuals, while Target Configuration Storage identifies and stores settings of existing config files and deploys them across multi-vendor network with produced templates.

organizations, even for simple tasks. This drift will be more severe in more complex configuration snippets [2].

However, we have discovered a common ground among vendors that they share similar high-level intents and corresponding parameter types, despite different configuration languages. It motivates our novel bottom-up CKB construction approach called ABC. By utilizing NLP models, we parse example snippets from device manuals to extract labeled parameters and generate configuration templates, without focusing on syntax variations in different configuration languages. Device manuals organize example snippets by intent along with their intent descriptions, enabling alignment of the produced templates to our unified configuration model (CCT in our work) of a certain intent. Additionally, our learned parameter extraction rules can extract labeled parameter settings from target configuration files, which are stored in CCT for achieving their high-level target intents across all devices.

C. Knowledge Extraction and ABC

ABC learns from device manuals to generate configuration templates and parameter extraction rules, which is similar to existing knowledge extraction tasks [11, 12, 17]. These tasks involve understanding the intent of an expression and performing Named Entity Recognition (NER) to identify entity types about that intent within the expression. For instance, given the intent expression “Block TCP traffic from A to B,” we identify the intent as *Block traffic* and then extract key information of that intent by recognizing traffic type, source, and destination nodes as *TCP*, *A*, and *B*. State-of-the-art approaches employ deep large language models like BERT [17] to efficiently handle intent understanding and NER using labeled data for intent types and corresponding entity types.

However, the knowledge extraction task in ABC is more challenging due to the extensive human effort needed to label intent and entity types for all vendors and intents in device manuals. To automate the task while maintaining accuracy, we propose an active-learning-based method with minimal expert involvement during training process instead of heavy pre-training labeling. We first just train one unified parameter extraction model using rough entity types (e.g. *PureNumber* and *IPorMask*) instead of refined types in a specific intent (e.g. *AS-number* and *SourceIP*) and form configuration templates with those rough types as parameter placeholders. By encoding templates with intent descriptions and comparing

their learning-based similarity, we group templates of the same intent across vendors and use minimal human effort in labeling the intent of the most puzzling templates for alignment to CCT. After alignment, specific entity types from CCT are used to refine the parameter extraction model for improved accuracy.

III. ABC DESIGN

In this section, we first present the workflow of ABC, then present a succinct introduction to technologies employed in its core modules and corresponding challenges.

A. System Overview

Fig. 2 shows that ABC has two workflows, the first is *ITC mapping* which involves learning NLP models for parameter parsing (PP) to generate configuration templates and template alignment based on intents and corresponding CCTs (ALIGN), while the second is *target configuration storing* which splits configuration files into intent-based snippets and extracts parameter settings from the snippets using trained models in PP.

The *ITC mapping* workflow is activated when a new vendor or intent is introduced and ABC fine-tunes the NLP models in PP and ALIGN module with the following steps:

- Device manuals are preprocessed to extract intent descriptions and example configuration snippets, forming intent-snippet tuples. The snippets serve as new training samples to fine-tune the NLP model in PP.
- The fine-tuned model in PP extracts parameters in example snippets and replaces them with their entity types as placeholders, creating snippet templates (see Fig. 3).
- Snippet templates replace the snippets in intent-snippet tuples, forming new training samples for ALIGN, which uses NLP models to encode intent descriptions and templates, grouping templates with the same intent together and aligning them to CCT with minimal human labeling.
- Template alignment causes a smaller entity set from CCT, thus we further refine parameter extraction NLP model in PP with reduced entity number for improved accuracy.

The *target configuration storage* workflow is triggered when a new configuration file of an arbitrary vendor or intent needs to be applied for all devices in the network. It has two main steps: (1) The snippet extraction module analyzes the configuration file and splits it into intent-based snippets. (2) Each intent-based snippet is processed by NLP models in the PP module to extract labeled parameters, which are then stored in corresponding CCT within the CKB.

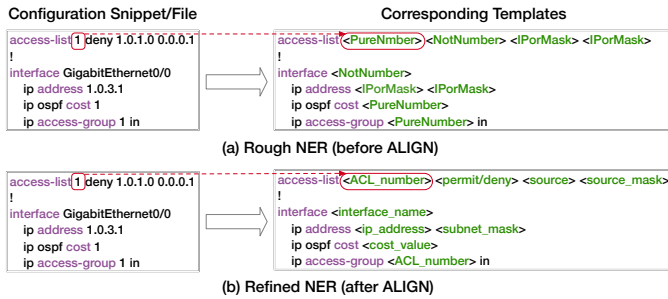


Fig. 3: Parameter extraction example. It involves a two-step NER task: (a) Using rough entity types, the NLP model generates initial templates for alignment procedure. (b) After alignment, specific entity types can be obtained from CCT to refine the model.

B. Preliminary Solutions and Challenges

We discuss our preliminary solutions and open questions of ABC’s core modules: PP, ALIGN and snippet extraction.

Parameter Parsing (PP). The PP module addresses the knowledge extraction task in ABC mentioned in Section II-C. It involves training an NLP model for NER tasks to extract parameters from example snippets in device manuals. This parameter extraction model is then used to generate configuration templates by replacing parameters with entity types as placeholders. However, due to the large number of network vendors and intents, it is impractical to have expert knowledge of every vendor and label intents and specific parameter entity types for all snippets in manuals. This limitation may result in providing insufficient labeled training samples for the NLP model, leading to degraded performance.

PP module in ABC aims to reduce the human effort required for labeling intent and parameter types in all intent-snippet tuples. We train a unified parameter extraction model using rough parameter entities (e.g., `PureNumber`) applicable across all intents and vendors (Fig. 3 (a)), instead of refined entities (e.g., `ACL_number` and `cost_value`) (Fig. 3 (b)). This rough NLP model parses the snippets into configuration templates. After aligning these vendor-specific templates with the CCT of a specific intent (ALIGN), we can determine a template’s specific parameter types from corresponding CCT. We then fine-tune the model to identify refined entity types based on rough entity types, requiring minimal human labeling of rough-refined entity mapping. We then use these fine-tuned parameter extraction models to generate more accurate configuration templates (Fig. 3 (b)). There are still open questions here to explore designing rough entity types and rough-refined mapping rules for efficient and accurate parameter extraction.

Templates Alignment (ALIGN). ALIGN groups configuration templates with similar intents from different vendors and aligns them with the corresponding CCT for specific intents and their parameter settings. However, human labeling intents for templates in various vendors is impractical for NetOps teams, hindering ALIGN’s goal with minimal human effort. Thus our proposed solution utilizes active-learning-based NLP encoders with little labeling process. ALIGN understands intent descriptions and their templates and groups templates with similar intents together by measuring their

similarity. During ALIGN’s learning pipeline, only a small set of representative templates are labeled and matched with their corresponding CCT by experts, reducing extensive pre-training labeling efforts.

Configuration Comprehension Model. We utilize fine-tuned large language models (LLMs) (i.e. BERT [17]) for encoding configuration templates and their intent descriptions, which involves parallel *Description* and *Template* encoders as shown in Fig. 4. Both encoders take the words in intent descriptions or templates as input and generate a *CLS* identifier each, representing the encoded sentences. We concatenate the *CLS* outputs of both encoders to obtain intent-template embeddings.

Training Process. To obtain accurate intent-template embeddings, we fine-tune LLMs using similarity-based training data and loss functions. Fig. 4 illustrates the training pipeline. First, we input all intent-template tuples into the encoders to obtain current embeddings. Then, we obtain the similarity matrix F by calculating normalized Euclidean distance between two arbitrary embeddings. By setting upper and lower thresholds for similarity, we determine which embeddings should be grouped together or not. This forms triple training data (P_i, q_{pos}, q_{neg}) , where P_i is the target embedding, q_{pos} represents embeddings that have the same intents as P_i and should be grouped, and q_{neg} represents embeddings not to be grouped. The loss function for the encoders aims to increase similarity between embeddings of the same intents and decrease similarity between different ones. It can be defined as:

$$\text{obj}(p_i, q_{pos}, q_{neg}) = \log(1 + e^{\text{dis}(p_i, q_{pos}) - \text{dis}(p_i, q_{neg})}), \quad (1)$$

where $\text{dis}(\cdot)$ represents the Euclidean distance.

Expert Labeling. The learning model alone may not efficiently identify similar intent templates without human labeling. To address this, ABC provides representative embeddings that puzzle similarity measurement for experts to label. For instance, experts label embeddings p_i and p_j when F_{ij} falls near the mean of the upper and lower thresholds. F_{ij} is labeled as 1 if the templates have the same intent and 0 otherwise. We also record the intent types of these labeled embeddings, for alignment of the grouped templates with corresponding CCTs.

Open Questions. As similarity-based alignment measurements do not provide ground truth, resulting in potentially incorrect training data generation even with expert labeling. Thus the alignment may not achieve 100% accuracy, which is not ideal for NetOps teams to maintain the CKB. Future works can focus on addressing this and exploring potential solutions.

Snippet Extraction. NetOps teams often need to extend settings in the existing configuration files of a specific vendor to the multi-vendor network. However, it is challenging because typical configuration files consist of mixed intents, while a single CCT corresponds to snippets of one intent. To address this, we propose to split configuration files into intent-based snippets to facilitate parameter storage in various CCTs. The straightforward approach to achieve this is to compare configuration files with snippet templates for all vendors and intents, but it is time-consuming and may require more human effort with complex scenarios such as mixed intents within the

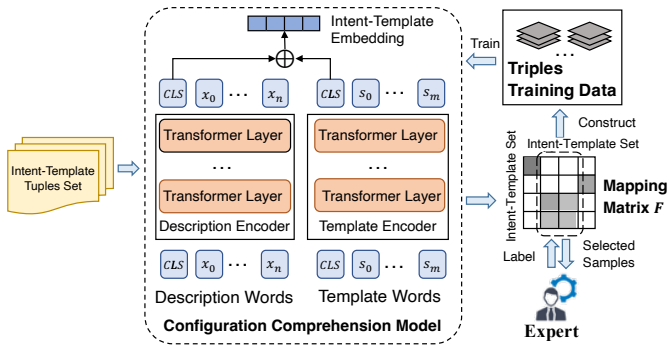


Fig. 4: The pipeline of ALIGN procedure. Two NLP encoders are employed to comprehend intents and their templates, while similarity-based training samples are used to group and align templates with similar intents.

same hierarchy (e.g., assigning OSPF weight and ACL of the same interface (see Fig. 3)).

To automate and accelerate this procedure, we primarily propose to construct a *statement tree* of the configuration file and enable *parallel matching* only with CCTs. The tree is built by organizing statements in configurations hierarchically with labeled parameters produced by PP module. Matching occurs between subtrees of the statement tree and CCTs in parallel, considering parameter types and successive node relationships. Processing from the root to the leaves, one subtree matching is terminated when mismatches occur. It remains an open question for a specific design to enhance accuracy and efficiency of this process.

IV. VISION FOR FUTURE WORKS

NetOps teams always seek to apply their high-level intents to the low-level configurations of different vendors with minimal human effort. ABC utilizes learning-based approaches to build a CKB, lightening the heavy expert workload for learning different configuration languages. ABC can serve as a milestone and foundation for the topic of *automated network configuration*. Future following works can be proposed to achieve a higher-level automation as outlined below.

Intent-based Network Configuration. Current CKB relies on existing configuration files to determine parameter settings and build shared configuration information bases across different vendors. However, we aim to achieve a more automated configuration approach by learning parameter settings directly from human intents, without writing any configuration file. To accomplish this, we develop tools that understand the intents of NetOps teams and translate them into specific parameter settings. Challenges may arise, such as dealing with indirect intents where parameters cannot be directly extracted from intents (e.g., modifying link weights for routing configuration), and detecting and resolving intent conflicts automatically where multiple intents provided by NetOps teams cannot be configured together in low-level implementation. Recent work in this direction, such as [10], primarily focuses on intent understanding and transferring intents into domain-specific language, rather than reaching the parameter settings level.

Network Configuration Synthesis. CKB is organized based on single intents. However, NetOps teams often propose

multiple intents for a network, necessitating efficient synthesis of these intent templates or snippets. Concatenating snippets with different intents is impractical due to the hierarchical organization of network configurations [5, 8]. For instance, configuring a port's feature may involve two different intents of assigning OSPF weights (routing configuration) and ACL, which reside in the same hierarchy. Recent research [15] also indicates that the configuration order of intent snippets affects the effectiveness of high-level intents. Consequently, it is challenging to develop network configuration synthesis tools based on intent-based CKBs with multiple high-level intents.

Intent-based Network Inquiry and Summarization. Existing network configuration files are often lengthy and written in low-level grammar, posing challenges for NetOps teams to comprehend and analyze network configuration status. ABC has introduced a network description encoder that understands human intents. By extending this module, NetOps teams can inquire snippets about specific intents and develop tools to summarize the associated configuration status of that intent. The summaries should be presented in an easily understandable format, such as natural language. Previous research [7] has proposed network summarization problem, yet only focusing on summarizing traffic information in forwarding tables. We aim to summarize configuration status of various intents across multiple configuration files based on human inquiries.

ABC Assisted Network Assimilation. A recent network configuration assimilation work Nassim [8] aims to automatically transfer network configuration files to other vendors but still uses top-down methods that require to handcraft vendor-specific parser functions. However, unlike ABC, Nassim provides well-designed parameter mapping solutions across different vendors while ABC leaves one-to-one mapping at the parameter level as an open question (see §III-B), only focusing on aligning snippet templates at the intent level. Combining the strengths of both approaches, ABC's bottom-up methods can be applied to Nassim, enabling the automatic construction of parser functions using learning models with device manuals. This integration may further reduce human efforts in Nassim.

V. CONCLUSION

NetOps teams always seek to implement low-level network configurations across different vendors using high-level intents, with minimal need for expert knowledge and human effort. Towards achieving this highly *automated network configuration* goal, the introduction of ABC will be a fundamental and significant contribution that builds CKB with snippet templates for consistent intent across vendors. It learns from device manuals and extracts labeled parameters from existing configuration files, enabling shared settings in a common knowledge base. Following this milestone, we can extend to consider more vision works that include solving parameter settings based on high-level intent sets and synthesizing configurations of different intents in a single file. We encourage the community to further contribute to ABC and to suggest more valuable follow-up works on this topic.

VI. ACKNOWLEDGMENT

This work is supported in part by funding from the Research Grants Council of Hong Kong (11209520) and from Huawei (CUHK #7010691).

REFERENCES

- [1] Apstra. <https://apstra.com/>.
- [2] Cisco OSs. https://www.reddit.com/r/Cisco/comments/2o619f/would_someone_please_explain_succinctly_what_the/cmkl1tma/.
- [3] Cisco Sues Huawei over Intellectual Property. <https://www.computerworld.com/article/2578617/cisco-sues-huawei-over-intellectual-property.html>.
- [4] Graph-Based Live Queries in AOS. <https://apstra.com/products/>.
- [5] OpenConfig. <http://openconfig.net/>.
- [6] Banerjee, Anubhab and Mwanje, Stephen S and Carle, Georg. Trust and Performance in Future AI-enabled, Open, Multi-Vendor Network Management Automation. *IEEE Transactions on Network and Service Management*, 20(2):995–1007, Oct. 2022.
- [7] Rüdiger Birkner, Dana Drachler-Cohen, Laurent Vanbever, and Martin Vechev. Net2Text: Query-Guided Summarization of Network Forwarding Behaviors. In *Proc. USENIX NSDI*, 2018.
- [8] Huangxun Chen, Yukai Miao, Li Chen, Haifeng Sun, Hong Xu, Libin Liu, Gong Zhang, and Wei Wang. Software-Defined Network Assimilation: Bridging the Last Mile Towards Centralized Network Configuration Management with NAssim. In *Proc. ACM SIGCOMM*, 2022.
- [9] Ari Fogel, Stanley Fung, Luis Pedrosa, Meg Walraed-Sullivan, Ramesh Govindan, Ratul Mahajan, and Todd Millstein. A General Approach to Network Configuration Analysis. In *Proc. USENIX NSDI*, 2015.
- [10] Arthur S. Jacobs, Ricardo J. Pfitscher, Rafael H. Ribeiro, Ronaldo A. Ferreira, Lisandro Z. Granville, Walter Willinger, and Sanjay G. Rao. Hey, Lumi! Using Natural Language for Intent-Based Network Management. In *Proc. USENIX ATC*, 2021.
- [11] Yoshihide Kato and Shigeki Matsubara. Parsing Gapping Constructions Based on Grammatical and Semantic Roles. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2020.
- [12] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Extracting Large-Scale Knowledge Bases from the Web. In *Proceedings of the International Conference on Very Large Data Bases*, 1999.
- [13] Hongqiang Harry Liu, Xin Wu, Wei Zhou, Weiguo Chen, Tao Wang, Hui Xu, Lei Zhou, Qing Ma, and Ming Zhang. NetCraft: Automatic Life Cycle Management of Network Configurations. In *Proc. ACM SelfDN*, 2018.
- [14] Martínez, Anny and Yannuzzi, Marcelo and López, Víctor and Lopez, Diego and Ramírez, Wilson and Serral-Gracia, Rene and Masip-Bruin, Xavi and Maciejewski, Maciej and Altmann, Jörn. Network Management Challenges and Trends in Multi-layer and Multi-vendor Settings for Carrier-grade Networks. *IEEE Communications Surveys & Tutorials*, 16(4):2207–2230, June 2014.
- [15] Tibor Schneider, Rüdiger Birkner, and Laurent Vanbever. Snowcap: Synthesizing Network-Wide Configuration Updates. In *Proc. ACM SIGCOMM*, 2021.
- [16] Yu-Wei Eric Sung, Xiaozheng Tie, Starsky HY Wong, and Hongyi Zeng. Robotron: Top-down Network Management at Facebook Scale. In *Proc. ACM SIGCOMM*, 2016.
- [17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, 2017.