Defining Operationality for Explanation-Based Learning

Richard M. Keller

Rutgers University Department of Computer Science
New Brunswick, New Jersey 08903
Internet: Keller@Rutgers.Edu

Abstract

Operationality is the key property that distinguishes the final description learned in an explanation-based system from the initial concept description input to the system. Yet most existing systems fail to define operationality with necessary precision. In particular, attempts to define operationality in terms of "efficient instance recognition" tacitly incorporate several unrealistic, simplifying assumptions about the learner's performance task and the type of performance improvement desired. Over time, these assumptions are likely to be violated, and the learning system's effectiveness will deteriorate. We survey how operationality is defined and assessed in several explanation-based systems, and then present a more comprehensive definition of operationality. We also describe an implemented system that incorporates our new definition and overcomes some of the limitations exhibited by current operationality assessment schemes.

I. Introduction

In recent years, the field of machine learning has experienced a surge of interest in a class of analytic concept learning methods called explanation-based methods [Mitchell *et al.*, 1986]. In contrast to empirical learning methods, which perform a simple syntactic analysis of similarities and differences among large numbers of training instances, explanation-based methods perform an in-depth, knowledge-intensive analysis of a single training example – typically a positive instance. That analysis involves first explaining why the positive training instance is an example of the concept to be learned (the *target concept*), and then generalizing the explanation in a principled manner so it is valid for a larger class of instances than the original instance. Finally, a description of that larger class of instances is extracted from the generalized explanation structure. The description constitutes a generalization of the original instance.

A seeming paradox of the explanation-based paradigm is that in order to produce its final description of the target concept, the learning system must possess an initial description of that same concept. In fact, without "knowing" an initial description of the target concept, it would be impossible for the system to explain why the given training instance is an example of the target concept. So if an initial target concept description is a prerequisite[1] for explanation-based methods, then why is learning necessary in the first place? What is wrong with the initial description? What is there to be learned? These questions are at the very heart of the "explanation-based paradox".

The way to untangle the paradox is to acknowledge that learning can involve *knowledge transformation*, as well as knowledge acquisition. Explanation-based methods do not acquire "new" knowledge, per se, but rather transform existing knowledge that is unusable or impracticable into a form that is usable [Keller, 1983, Dietterich, 1986]. In particular, although the initial target concept description given to an explanation-based system generally is correct (i.e., it covers the correct set of instances), the description is in *non-operational* form. Informally, this means the description cannot be used effectively by the learner to improve task performance. There is a significant difference between having a concept description and being able to use it; the task of an explanation-based system is to narrow that gap by transforming, or operationalizing, the initial description.

As a concrete example for illustration, consider Winston et al.'s analogy system, which uses explanation-based methods to learn a description for CUP [Winston *et al.*, 1983]. In this case, an initial CUP description is given to the system, expressed in functional terms: "a CUP is an open, stable, liftable vessel." This description is non-operational because it does not contain the necessary information to enable a vision system (the performance system) to improve its performance in recognizing CUPs. In order for the description to be useful in improving performance, the learning system must transform the functional description into a *structural* description, composed of primitives the vision system is designed to recognize: "a CUP is a light object with a handle, a flat bottom, and an upward-pointing concavity."

Although operationality is the key property that distinguishes the final description learned in an explanation-based system from the initial target concept description, most existing systems fail to define operationality with necessary precision. Current attempts to define operationality

[1] The initial target concept description may be represented in the learning system explicitly (i.e., in terms of declarative structures) or implicitly (i.e., within the system's procedures) [Keller, 1987b].

tacitly incorporate several unrealistic, simplifying assumptions that may not hold throughout the course of learning. In particular, many explanation-based systems treat operationality as an independent, static, binary-valued property of concept descriptions. Actually, operationality is a dynamic, continuous-valued property of descriptions, dependent on the learner's performance task and the type of performance improvement desired. A more thorough understanding of operationality is necessary to construct sophisticated explanation-based systems that can function properly in dynamic, real-world environments.

In what follows, we define operationality more precisely and analyze how operationality is assessed in several explanation-based systems. Then we describe how the MetaLEX system [Keller, 1987b, Keller, 1987a] overcomes some of the limitations exhibited by current operationality assessment schemes, by incorporating our new definition of operationality.

II. Preliminaries

This section introduces some terminology and establishes a common framework to serve as a basis for our discussion of operationality.

We begin by distinguishing between a concept and its description. A *concept* represents a subset of instances drawn from some universe. A concept is denoted intensionally by a *concept description*, which is a predicate over the universe of instances. Two concept descriptions are considered *synonymous* if they denote the same concept. Figure 1 illustrates these relationships. In the center, the figure depicts the space of all possible concepts that can be described by a given learning program. Each point in concept space represents a unique set of instances drawn from instance space, shown at right. At left, the figure depicts the space of all possible descriptions of the concepts in concept space. The description space is partitioned into operational and non-operational regions. Note that there is a one-to-many correspondence between a point in concept space and the points in concept description space. As illustrated, D_1 and D_2 are two synonymous descriptions, both describing concept C, which covers instances I_1, I_2, and I_3. However D_2 is considered operational, whereas D_1 is not. So when there exist different ways to describe the same concept, operationality defines the criterion for preferring one description over another.

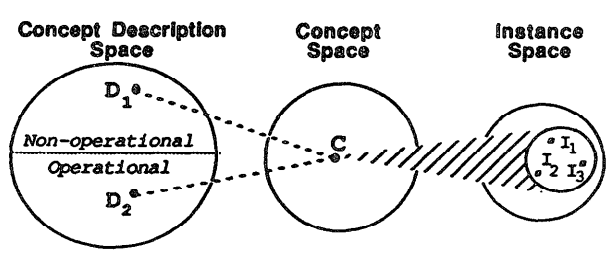

Figure 1: A Concept and its Descriptions

The role of operationality with respect to explanation-based learning is clarified by viewing explanation-based learning as a search through the concept description space. Suppose D_1 in Figure 1 is the initial, non-operational description provided to an explanation-based system, and D_2 is the final, operational description learned. Then we can view D_1 as the starting node in a search, D_2 as a solution node, explanation as the means for traversing the space, and operationality as the search termination criterion. We call the process of transforming D_1 into D_2 "concept operationalization" [Keller, 1987b, Keller, 1983].

Mitchell first described learning as a search process [Mitchell, 1982]. A crucial distinction between his formulation of the problem and ours is that Mitchell effectively equates a concept with its description, thereby masking the issue of operationality. Thus he effectively characterizes learning as a search through a concept space, not a concept *description* space. This characterization is insufficient for describing explanation-based learning. Explanation-based learning involves *no* searching through the concept space because the initial description (D_1) and the final description (D_2) denote the same concept (C).

III. Defining Operationality

The definition of operationality most commonly cited in describing explanation-based systems is the following:

> **Current Operationality Defn.:** A concept description is *operational* if it can be used efficiently to recognize instances of the concept it denotes.

Below we review how this definition is instantiated in several systems that use explanation-based methods. Note that operationality has not been defined explicitly in several of these systems, so the following definitions are based on our retrospective analysis and reconstruction.

- **Winston et al.'s Function-Structure System** [Winston *et al.*, 1983]: In this system, the target concept is the set of drinking CUPs and the initial description is a *functional* description: "a CUP is any open, stable, liftable vessel." Instances, however, consist of physical descriptions of CUPs from the real world, expressed in terms of *structural* properties, such as "flat", "handle", "concave", etc. Therefore, an operational CUP description is defined in this system as a description stated solely in structural terms, so it can be easily matched against instances.
- **LEX2** [Mitchell *et al.*, 1986]: In LEX2, the target concept is the set of USEFUL problem solving moves to apply during search. The initial description of USEFUL given to LEX2 states that "USEFUL moves lead immediately or eventually to a solution." Instances consist of calculus problem solving moves generated while solving actual problems. To facilitate matching against instances, an operational description of USEFUL is defined in LEX2 as a description expressed in terms of the calculus features used to describe instances (e.g., "sin", "3", "product", etc.), or in terms of features easily derivable from them (e.g., "trig-function", "integer", "polynomial", etc.).

• **PRODIGY** [Minton, 1986]: This system learns a variety of target concepts related to problem solving, including the USEFUL concept learned by LEX2. One of PRODIGY's problem solving domains is a machine-shop scheduling domain, in which raw materials are transformed into finished goods using operators like LATHE, CLAMP, POLISH, etc. PRODIGY, for example, can learn conditions under which applying these operators is UNSUCCESSFUL. Instances correspond to different states of the machine-shop environment in which the operators are applied. An operational description must be phrased in terms of directly observable features of the raw materials and the machine-shop equipment in the environment, such as "shape", "temperature", "idle", "busy". The direct observability requirement assures that UNSUCCESSFUL conditions can be recognized quickly and operator application can be avoided in a real-time environment.

• **GENESIS** [Mooney and DeJong, 1985]: In one of GENESIS's application domains, the target concept corresponds to WEALTH-ACQUISITION-SCENARIO, and an initial (although not explicitly stated) description of the target concept is that "a WEALTH-ACQUISITION-SCENARIO consists of any sequence of actions that culminate in an agent's acquisition of wealth." Instances consist of natural language text describing stories involving acquisition of wealth, such as stories involving inheritance, kidnapping, arson, etc. Unlike in the three systems described above, an operational description for GENESIS is *not* stated in terms of the low-level features present in the instances, but rather in terms of abstract schemata possessed by the system. A high-level description of WEALTH-ACQUISITION-SCENARIO facilitates "efficient instance recognition" because the story understanding component in GENESIS parses stories (i.e., instances) efficiently in top-down fashion.

• **SOAR** [Rosenbloom and Laird, 1986]: In SOAR, a form of explanation-based learning is an integral part of its chunking mechanism. Each time SOAR completes problem solving activity for a specific subgoal, the system attempts to construct a generalized production to achieve "similar" subgoals without resorting to problem solving. For our purposes, we can consider a specific subgoal (along with the current processing state) as a training instance, the class of "similar" subgoals as SOAR's target concept, and the generalized production's conditions as its operational description. An operational description of the target concept is one that can be used to efficiently recognize whether a "similar" subgoal is true in a given processing state. In other words, in SOAR an operational production consists solely of conditions that can be easily evaluated without problem solving, including conditions initially present in SOAR's working memory, and conditions that are evaluated using chunks acquired during problem solving.

The notion of "efficient instance recognition" employed in these systems is a suitable starting point for defining operationality, but is in several ways inadequate as a general-purpose definition. A basic problem is that the "efficient instance recognition" definition tacitly incorporates several restrictive assumptions about how the final concept description will be used to improve performance.

First, the definition assumes that the concept description will be used to "recognize instances". Although instance recognition represents one typical use of a concept description, there are other uses, including instance *generation*. For example, in the CUP domain, we might be interested either in *recognizing* cups (e.g., if we want to drink a beverage) or in *generating* cups (e.g., if we are designing new types of cups). An operational description for the purposes of generation is *functional*, rather than structural, because a larger number of novel cup designs can be generated from the abstract functional description.

Second, the "efficient instance recognition" definition used by most systems assumes that execution time is the proper measure of performance to use in evaluating operationality. However, there are other types of "efficiency" that may be just as appropriate or more appropriate for evaluating performance, including space efficiency. A description that is operational with respect to time efficiency may not be operational with respect to space efficiency. Furthermore, aside from efficiency, there are arbitrarily large numbers of other criteria that might be relevant to performance, including cost, elegance, simplicity, etc.

The way to eliminate these restrictive assumptions from the definition of operationality is to redefine it in terms of the performance system that uses the learned concept description, and in terms of the criteria for evaluating that system's performance. Table 1 gives our revised definition of operationality. There are two requirements on operationality in the revised definition: *usability* and *utility*. The usability requirement ensures that the description can be used by the performance system. This means that the description must be expressed in terms of capabilities possessed by the system, and in terms of data known or computable by the system. (The usability requirement corresponds to the original notion of operationality introduced in [Mostow, 1981].) The utility requirement takes usability one step further: the description must not only

Table 1: **Revised Operationality Definition**

Given:

- A *concept description*
- A *performance system* that makes use of the description to improve performance
- *Performance objectives* specifying the type and extent of system improvement desired

Then: the concept description is considered **operational** if it satisfies the following two requirements:

1. *usability*: the description must be usable by the performance system
2. *utility*: when the description is used by the performance system, the system's performance must improve in accordance with the specified objectives.

be usable, but also *worth* using. In particular, using the description must improve the behavior of the performance system, as defined by its performance objectives.

As an example of how this revised operationality definition might be instantiated for existing explanation-based systems, consider once again Winston et al.'s CUP domain. In this domain, the performance system might consist of a mobile robot searching for cups in a room. The performance objectives for the robot might be to improve the speed with which it can recognize and retrieve cups. For a CUP description to be "usable" by the robot, it must be expressed in terms of object properties that can be detected by the robot's sensory systems. Those properties correspond to structural properties. For a CUP description to be "utile", as well as "usable", the robot must be able to easily evaluate the properties used in the description. Therefore, a structural property such as "specific-gravity", for instance, would not be permitted as part of an operational description.

With the revised definition, the notion of operationality adjusts to fit the learning situation. Continuing with the above example, suppose instead we are learning about cups in a design context. In this case, the performance system might consist of a design system containing a library of functional design primitives. The performance objective might be to increase the number of cup designs the system can generate. Now the revised operationality definition correctly pinpoints an operational description as one expressed in terms of the *functional* design primitives known by the system, instead of structural primitives.

IV. Assessing Operationality

In this section, we discuss how operationality is assessed in explanation-based systems. Thus we draw a distinction between how operationality is defined and how it is evaluated in practice. Conceptually, each explanation-based system contains an *operationality assessment procedure*, which evaluates a concept description and produces a measure of its operationality as output. Below, we describe three dimensions – *variability*, *granularity*, and *certainty* – which characterize the operationality measurements produced by an assessment procedure. A comparison of various systems' assessment procedures along these dimensions is given in Table 2. (The table includes the systems described in the previous section, as well as the MetaLEX system, which is described in the next section.)

Table 2: **Characterizing Operationality Assessment**

Granularity	*Variability*	
	static	dynamic
binary	Winston†	GENESIS† SOAR† LEX2†
continuous	PRODIGY†	MetaLEX*

† *Certainty*: unguaranteed * *Certainty*: guaranteed

• **Variability**: A dimension characterizing whether operationality assessment varies with time. Values: *static* or *dynamic*.

As learning progresses, a description that is initially non-operational may become operational, and vice versa, due to changes in the performance environment. An accurate assessment of operationality depends on *when* the assessment is made. For example, consider a performance system consisting of a mobile robot equipped with a black and white camera. For this system, any object descriptions which specify color attributes should be considered *non-operational* for recognition. However, if the camera were replaced with a color camera, these same descriptions should be considered *operational* for the updated system.

Some of the systems surveyed in the previous section perform a dynamic assessment of operationality, whereas others do not. Assessment in GENESIS and SOAR is dynamic because operationality is defined in terms of the *existing* set of schemata or chunks, respectively. As these systems acquire additional schemata or chunks, the set of descriptions considered operational is enlarged. Similarly, the set of operational descriptions in LEX2 is augmented when the STABB subsystem [Utgoff, 1986] adds a new term to the system's generalization language. Note, however, that the set of operational descriptions in Winston et al.'s system and in PRODIGY remains static throughout the course of learning, so these systems cannot automatically adjust to changes in the performance environment.

• **Granularity**: A dimension characterizing the assessment measure produced. Values: *binary* or *continuous*.

Most of the systems surveyed produce a binary assessment of operationality: either "operational" or "non-operational". However, continuous-valued assessment has distinct advantages over binary assessment because it allows the learning system to assess *degrees* of operationality. In situations where there exist several synonymous, operational descriptions of the target concept, a metric on operationality enables the system to learn the "best" (i.e., most effective) description. Additionally, continuous-valued assessment facilitates attempts to guide the search through concept description space by providing a measure of progress through the space.

PRODIGY is one system that features continuous-valued assessment. In other words, PRODIGY can assess *how* efficient a given description is for the purposes of recognition. The system bases this assessment on an *a priori* estimate of the matching costs associated with each "observable" feature in the machine-shop environment. Given two synonymous, operational descriptions of the target concept, PRODIGY evaluates which is more operational using the matching cost estimates.

• **Certainty**: A dimension characterizing confidence in the measurement produced by the operationality assessment procedure. Values: *unguaranteed* or *guaranteed*.

None of the systems surveyed can guarantee the accuracy of their assessment measurements, because none of the systems directly assess operationality. In other words, the

systems do not actually test whether a given description is "efficient to use for recognizing instances". Instead, the systems base their assessments on whether the description being assessed is contained within a pre-defined *language of operational descriptions*, which is specified by the learning system's human designer. This language includes only terms that the designer decides can be evaluated efficiently by the performance system in order to accomplish instance recognition. In a sense, the language is a "compiled" form of the designer's knowledge about the performance system [Keller, 1987b]. For LEX2, that language includes descriptions expressed in terms of a variety of "syntactic" features of calculus problem solving states. For GENESIS, the language includes any description composed of schemata possessed by the system.

The problem with using a "compiled" language to assess operationality, is that the original performance assumptions upon which the designer based the language may become invalid. In fact, this is inevitable as the performance system's capabilities change over time. The result is that the original language definition no longer correctly identifies operational descriptions.

Consider an example from SOAR for clarification. As discussed in the previous section, SOAR's operational language consists of descriptions involving production conditions that have already been chunked, and thus are presumed efficient to use in recognition. However, SOAR's problem solving behavior will likely deteriorate over time as more chunks are learned and matching costs increase. (This type of difficulty has been documented – with plans, rather than chunks – for STRIPS-type problem solvers [Minton, 1985].) Eventually, descriptions involving any arbitrary chunked condition will no longer be efficient to use. At this point, it may be necessary to redefine operationality so that only the "most efficient" chunks are permitted within operational descriptions. But SOAR lacks the proper perspective over its own problem solving/learning behavior to identify that the operational language definition has become invalid over time. Moreover, SOAR cannot automatically "recompile" a new language definition, so changing the definition requires human intervention.

As is evident from Table 2, the MetaLEX program is in a different equivalence class with respect to these three dimensions. The next section discusses MetaLEX and its treatment of operationality.

V. MetaLEX

The MetaLEX program [Keller, 1987b, Keller, 1987a] is a successor to LEX2, designed to explore the concept operationalization paradigm introduced in [Keller, 1983]. MetaLEX approaches essentially the same learning task as LEX2, but solves it using a different technique. In particular, both MetaLEX and LEX2 learn a description of the set of USEFUL problem solving moves to execute during forward search. Both systems start with the same non-operational target concept description ("a USEFUL problem solving move leads eventually to a solution state"), and attempt to transform it into an operational description. Where the systems differ is in their methods for accomplishing the transformation.

LEX2 transforms the initial target concept description using explanation-based methods. In a sense, LEX2 conducts a bi-directional search of the concept description space, with the initial target concept description anchoring the search in the non-operational region and a training instance description anchoring the search in the operational region. The explanation is used as a means of traversing the search space.

In contrast, MetaLEX searches out from the initial description in the direction of an operational description using a form of hill-climbing. We do not describe the hill-climbing algorithm or the operators used to search the space in this paper. Instead, we focus on how MetaLEX assesses the operationality of a concept description.

The definition of operationality used by MetaLEX is given in Table 3. Note that there are two objectives specified for the SOLVER performance system, one involving efficiency and the other involving effectiveness. The objectives require that an operational description improve SOLVER's problem solving efficiency while also maintaining its effectiveness. The efficiency and effectiveness performance measures establish a metric over the description space that guides the hill-climbing search.

To assess operationality for a description of the USEFUL concept, MetaLEX inserts that description into SOLVER and observes SOLVER's behavior on a set of benchmark calculus problems. During SOLVER's execution, the USEFUL description guides expansion of problem solving moves: any move recognized as USEFUL is expanded, while other moves are pruned. SOLVER's efficiency and effectiveness are monitored during the problem solving session. If SOLVER's performance meets the

Table 3: MetaLEX Operationality Definition

Given:

- *Concept description:* Description of class of USEFUL problem solving moves to expand during search
- *Performance system:* SOLVER, a forward search problem solving system
- *Performance objectives:* Improve SOLVER's run-time efficiency on a set of benchmark calculus problems by X%, while maintaining its effectiveness in solving those problems correctly

Then: the USEFUL description is considered operational if it satisfies the following two requirements:

1. *usability*: the description is usable (i.e., evaluable) by SOLVER and
2. *utility*: using the description, SOLVER's efficiency achieves an X% improvement without a deterioration in effectiveness.

established performance objectives, the USEFUL description is considered operational. If SOLVER's performance fails to attain the desired levels of efficiency or effectiveness, MetaLEX can evaluate how far from operational the description is, and can assess whether the operationalization search is headed in the right direction.

In the terminology of the previous section, MetaLEX's operationality assessment method can be characterized as:

- "dynamic" – because operationality assessment depends on the current state of SOLVER and the current performance objectives;
- "continuous" – because assessment yields a measure of the *degree* of efficiency (in CPU seconds) and the *degree* of effectiveness (in number of benchmark problems solved); and
- "guaranteed – because assessment is accomplished by directly executing SOLVER, and observing whether its performance meets the stated objectives.

However, MetaLEX's assessment method is also extremely expensive because the performance system must be tested each time operationality assessment is required. MetaLEX reduces these costs by estimating system performance whenever possible in lieu of executing a system test.

VI. Conclusion

Operationality is the key feature that distinguishes the output concept description from the input concept description in an explanation-based system. As such, operationality is at the heart of what it means for an explanation-based system (or more generally, for a knowledge transformation system) to "learn." Yet current methods for assessing operationality tacitly depend on simplifying performance assumptions that likely will be violated as learning progresses. The MetaLEX program circumvents the problems associated with violated assumptions by defining operationality directly in terms of the performance system. The handling of operationality in MetaLEX may provide insight into how to construct general purpose explanation-based learning systems — systems that are more sophisticated in their operationality assessment capabilities, and that function properly over time, and for tasks other than "efficient instance recognition.

VII. Acknowledgments

Thanks go to my thesis advisor, Tom Mitchell, and to Jack Mostow, both of whom provided invaluable guidance in directing this research. Smadar Kedar-Cabelli provided encouragement and helpful comments on earlier drafts of this paper. Discussions with Steve Minton helped clarify the presentation of PRODIGY and SOAR. Chun Liew provided LaTeX formatting assistance. Funding to support this research was provided by NSF grant #DCS83-51523 and DARPA contract #N00014-85-K-0116.

References

[Dietterich, 1986] T. G. Dietterich. Learning at the knowledge level. *Machine Learning*, 1(3):287–316, 1986.

[Keller, 1983] R. M. Keller. Learning by re-expressing concepts for efficient recognition. In *Proceedings AAAI-83*, pages 182–186, Washington, D.C., August 1983.

[Keller, 1987a] R. M. Keller. Concept learning in context. In *Proc. 4th International Machine Learning Workshop*, University of California, Irvine, June 1987.

[Keller, 1987b] R. M. Keller. *The Role of Explicit Contextual Knowledge in Learning Concepts to Improve Performance.* PhD thesis, Rutgers University, January 1987. Technical Report #ML-TR-7.

[Minton, 1985] S. Minton. Selectively generalizing plans for problem-solving. In *Proceedings IJCAI-9*, pages 596–599, Los Angeles, CA, August 1985.

[Minton, 1986] S. Minton. Improving the effectiveness of explanation-based learning. In *Proceedings of the Workshop on Knowledge Compilation*, Oregon State University, Corvallis, September 1986.

[Mitchell, 1982] T. M. Mitchell. Generalization as search. *Artificial Intelligence*, 18(2):203–226, March 1982.

[Mitchell *et al.*, 1986] T. M. Mitchell, R. M. Keller, and S. T. Kedar-Cabelli. Explanation-based generalization: a unifying view. *Machine Learning*, 1(1), 1986.

[Mooney and DeJong, 1985] R. Mooney and G. DeJong. Learning schemata for natural language processing. In *Proceedings IJCAI-9*, pages 681–687, Los Angeles, CA, August 1985.

[Mostow, 1981] D. J. Mostow. *Mechanical Transformation of Task Heuristics into Operational Procedures.* PhD thesis, Comp.Sci.Dept., CMU, 1981.

[Rosenbloom and Laird, 1986] P. S. Rosenbloom and J. E. Laird. Mapping explanation-based generalization onto soar. In *Proceedings AAAI-86*, AAAI, Philadelphia, PA, August 1986.

[Utgoff, 1986] P. E. Utgoff. *Machine Learning of Inductive Bias.* Kluwer Academic, Hingham, MA, 1986.

[Winston *et al.*, 1983] P. H. Winston, T. O. Binford, B. Katz, and M. Lowry. Learning physical descriptions from functional definitions, examples, and precedents. In *Proceedings AAAI-83*, pages 433–439, Washington, D.C., August 1983.