УДК 004.41

ПРИМЕНЕНИЕ ПОРОГОВЫХ ФУНКЦИЙ ДЛЯ ИЗВЛЕЧЕНИЯ ИНФОРМАЦИИ О ТИПАХ ДАННЫХ ПРИ ПОЛУЧЕНИИ ПРОМЕЖУТОЧНЫХ ПРЕДСТАВЛЕНИЙ ТЕКСТОВ ПРОГРАММ ДЛЯ ЯЗЫКОВ С ДИНАМИЧЕСКОЙ ТИПИЗАЦИЕЙ

Зубов М.В., Пустыгин А.Н., Старцев Е.В.

В статье описывается подход, позволяющий получить информацию о типах данных в языках динамического типа с использованием пороговых функций. Описана математическая модель подхода и приведены результаты применения методики для ряда проектов, демонстрирующие ее эффективность.

Ключевые слова: статический анализ, промежуточное представление, исходный код, динамическая типизация, математическое моделирование.

Введение

В [1] был предложено решение проблемы, характерной для статического анализа динамических языков программирования, таких как Python, поскольку информация о типах имен не может быть получена в явном виде из исходного кода. Метод решения был основан на подборе классов-кандидатов для полей, называемых в методике утиными: сигнатура утиного поля сравнивалась с характеристической сигнатурой возможного класса-кандидата. Характеристическая сигнатура класса-кандидата представляла собой кортеж, содержащий множества его полей и методов, а сигнатура утиного поля – полей и методов, к которым происходило обращение в методах класса, содержащего утиное поле.

Ограничения первоначальной методики

Класс выбирался в качестве кандидата, если множество методов его сигнатуры являлось подмножеством множества методов сигнатуры утиного поля, а множество полей его сигнатуры являлось подмножеством множества полей сигнатуры утиного поля. Мы предполагаем, что взаимодействия с полями и методами утиного поля внутри методов класса, которому он принадлежит, происходит посредством интерфейса некоторого суперкласса, то есть каждое обращение к полям и методам утиного класса – это обращение к полям и методам, характерным для данного суперкласса.

Это довольно жесткое ограничение, так как поле класса при разных сценариях исполнения программы может хранить объекты различных классов, интерфейс работы с которыми не ограничивается сигнатурой некоторого суперкласса. Эти классы вполне могут вообще не быть наследниками некоторого общего суперкласса, либо помимо полей и методов суперкласса будут также использоваться поля, характерные только для наследников. В таком случае в проекте может не быть класса, в сигнатуре которого есть все поля и методы из сигнатуры утиного поля, так как в нее входят поля и методы различных классов.

Математическая модель предлагаемой усовершенствованной методики

Предлагается ввести пороговую функцию оценки класса в качестве кандидата для определенного утиного поля. Функция принимает значения от 0 до 1, формирует количественную оценку класса в качестве типа утиного поля. 1 соответствует максимальной оценке, а 0 – минимальной. Класс выбирается в качестве кандидата, если значение этой функции для него больше определенного порога.

Согласно терминологии [1], отношение $\mathrm{D_D}$ содержит пары классов и их утиных полей. Отметим, что утиными полями в предлагаемой методике являются те поля классов, которые как-либо использовались в методах класса, которому они принадлежат. Под использованием в данном случае понимаем обращение к полям или методам этого поля. Также введем отношения

$$\mathrm{Y_f} \subseteq \mathrm{C} \times \mathrm{F} \times \mathrm{F},\ \mathrm{Y_m} \subseteq \mathrm{C} \times \mathrm{F} \times \mathrm{M},$$

определяющие все поля и методы, соответственно, утиных полей, к которым происходило обращение

$$\mathrm{Y_f} = \left\{ \left(c, f, f_d\right) \middle| \left(c, f\right) \in \mathrm{D_D}, f_d \in \mathrm{F} \right\},$$

где имеется обращения к полю f_d утиного поля f класса *c*, и

$$\mathrm{Y_m} = \left\{ \left(c, f, m\right) \middle| \left(c, f\right) \in \mathrm{D_D}, m \in \mathrm{M} \right\},$$

где имеется обращение к методу *m* утиного поля f класса *c*.

Рассмотрим получение этих отношений для листинга 1. Для класса Document Handler из листинга 1 утиными полями, используемыми в ме-

тодах класса, являются поля _doc и _handler, для первого из этих полей происходило обращение к методу update_links и полям head, contents и index, а для второго обращение к методам update_head, protect_contents, get_str.

Листинг 1. Пример класса для применения методики типизации

```
class DocumentHandler(object):
  _doc = None
  _handler = None
  def __init__(self,factory,doc_factory):
    self._handler = factory.get_handler()
    self._doc = doc_factory \
                .get_current_doc()

  def change_head(self,new_head)
    self._doc.head = new_head
    self._doc.update_links()
    self._handler.update_head(self._doc)

  def get_contents_by_head(self,head):
    if(self._doc.head==head):
      self._handler \
          .protect_contents(self._doc)
    return self._doc.contents

  def find(self,name):
    for n in self._doc.index:
      if(n.name==name):
        return self._handler \
                    .get_str(n.ids)

  def find_and_replace(self,name, \
                       changed_name):
    for n in self._doc.index:
      if(n.name==name):
        for s in self._handler \
                     .get_str(n.ids):
          s.replace(name,changed_name)
```

В данном случае:

$$\mathrm{Y_f} = (DocumentHandler, \_\mathrm{doc}, \mathrm{head}),$$
$$(DocumentHandler, \_\mathrm{doc}, \mathrm{contents}),$$
$$(DocumentHandler, \_\mathrm{doc}, \mathrm{index}).$$
$$\mathrm{Y_m} = (DocumentHandler, \_\mathrm{doc}, \mathrm{update\_links}),$$
$$(DocumentHandler, \_\mathrm{handler}, \mathrm{update\_head}),$$
$$(DocumentHandler, \_\mathrm{handler}, \mathrm{protect\_contents}),$$
$$(DocumentHandler, \_\mathrm{handler}, \mathrm{get\_str}).$$

Возможные варианты пороговой функции.

1. Учитывать только процент полей из полной сигнатуры утиного поля, которые имеются у данного класса. Класс будет подобран в качестве кандидата, если у него имеется определенная доля полей и методов из сигнатуры утиного поля. Будем использовать для обозначения данной функции название «capacity».

2. Учитывать частоту использования полей в методах класса, наибольшую долю в итоговой оценке отдавать тем полям, которые использовались наиболее часто у класса. Частота использования – достаточно важный параметр для принятия решения о потенциальном классе-кандидате. Ведь поле, которое использовалось трижды, несет гораздо больше информации о потенциальном классе-кандидате, чем поле, к которому обращались лишь один раз. Будем использовать для обозначения данной функции название «frequency».

Пороговая функция «capacity»

В первом случае в качестве оценочной функции будет использоваться отношение общего количества полей и методов из сигнатуры утиного поля, которые имеются у рассматриваемого класса, к общему количеству полей и методов из сигнатуры утиного поля.

Сигнатура класса $\mathrm{c_C} \in \mathrm{C}$ – $\mathrm{s_C} = (\mathrm{F_C}, M_C), F_C \subseteq F, M_C \subseteq \mathrm{M}$, сигнатура утиного поля класса $\mathrm{f} \in \mathrm{F}$ – $\mathrm{s_D} = (\mathrm{F_D}, M_D), F_D \subseteq F, M_D \subseteq \mathrm{M}$. Для поиска кандидатов вводится пороговая функция:

$$\chi(\mathrm{c_S}, f, c_C) = \frac{|\mathrm{F_C} \cap \mathrm{F_D}| + |\mathrm{M_C} \cap \mathrm{M_D}|}{|\mathrm{F_D}| + |\mathrm{M_D}|}. \quad (1)$$

При использовании такой функции высокую оценку будут получать те классы, которые содержат больше различных полей и методов, характерных для сигнатуры утиного класса. Максимальная оценка будет у тех классов, которые содержат все поля и методы из утиной сигнатуры. Классы, которые подбирались как кандидаты для типа поля в соответствии с оригинальной методикой, соответствуют 1 значению этой функции, так как в оригинальной методике все поля и методы из сигнатуры утиного поля должны были входить в сигнатуру класса, который выбирался в качестве класса-кандидата.

Пороговая функция «frequency»

Введем функции $\tau_f(c, f_D, f), (c, f_D) \in \mathrm{D_D}$ и $\tau_\mathrm{m}(c, f_D, m), (c, f_D) \in \mathrm{D_D}$, которая для каждого используемого утиного поля f_D класса с возвращает количество случаев обращения к полю f и методу m, соответственно.

Для подсчета общего количества использования полей и методов всех утиных полей класса можно ввести функцию

$$\lambda(c,f)=\sum_{(c,f,f_i)\in Y_f}\tau_f(c,f,f_i)+$$
$$+\sum_{(c,f,m_i)\in Y_m}\tau_m(c,f,m_i),(c,f)\in D_D,$$

которая для каждого класса возвращает общее количество случаев обращения к полям/методам его утиных полей.

Тогда для каждого поля f_i и метода m_i утиного поля f класса c можно соответственно определить оценочные функции:

$$\delta_f(c,f,f_i)=\frac{\tau(c,f,f_i)}{\lambda(c,f)};\ \delta_m(c,f,m_i)=\frac{\tau(c,f,m_i)}{\lambda(c,f)}.$$

Чем больше будет обращений к определенному методу или методу утиного поля, тем выше будет значение данной функции для него. Сумма значений данной функции для всех полей и методов утиного поля класса, очевидно, равно 1.

$$\sum_{(c,f,f_i)\in Y_f}\delta_f(c,f,f_i)+\sum_{(c,f,m_i)\in Y_m}\delta_m(c,f,m_i)=1,$$
$$\forall(c,f)\in D_D.$$

Определим пороговую функцию $\omega(c_C,f,c_S)$ для определения класса $c_C\in C$ как кандидата для утиного поля $f\in F$ класса $c_S\in C$. Сигнатура класса $c_C\in C$:

$$s_C=(F_C,M_C),F_C\subseteq F,M_C\subseteq M;$$
$$\omega(c_S,f,c_C)=$$
$$=\sum_{f_i\in F_C,(c_S,f,f_i)\in Y_f}\delta_f(c_S,f,f_i)+$$
$$+\sum_{m_i\in M_C,(c_S,f,m_i)\in Y_m}\delta_m(c_S,f,m_i). \quad (2)$$

Функция (2) суммирует значения функции $\delta_f(c,f,f_i)$ и $\delta_m(c,f,m_i)$ для тех утиных полей и методов, соответственно, класса $c_S\in C$, которые также имеются у класса $c_C\in C$.

Для решения задачи поиска кандидатов для типов утиных полей вводится некоторый порог p. Условием того, что класс учитывается как кандидат для типа утиного поля, будет превышение значения пороговой функции данного утиного поля для этого класса. Если обозначить за $p1$ и $p2$, соответственно, пороги для первого и второго подходов, то можно записать эти условия как

- алгоритм поиска кандидатов для типов утиных полей на основе пороговой функции «capacity»:

c_C кандидат для поля f класса $c_S\Leftrightarrow$

$$\Leftrightarrow\chi(c_C,f,c_S)>p1;$$

- алгоритм поиска кандидатов на основе пороговой функции «frequency»:

c_C кандидат для поля f класса $c_S\Leftrightarrow$

$$\Leftrightarrow\omega(c_C,f,c_S)>p2.$$

Результаты применения методики

Основные результаты дает проверка методики на основе динамического анализа на типовых сценариях исполнения для проектов. Детали динамической методики были рассмотрены в [1]. Суть способа в том, что благодаря динамическому анализу получается корректная информация о типах данных полей классов. Эта информация сравнивается с теми результатами, которые получились при статическом анализе. Сценарии исполнения для трех типовых проектов: logilab [2], pylint [3] и bazaar [4] описаны в [1].

Особый интерес при динамической проверке представляет исследование влияния используемого порога для функции при подборе классов-кандидатов. Использовалось десять различных значений порога: от 0,1 до 0,95 для обеих пороговых функций. Значения порога 1.0 не исследовались, так как соответствуют подбору класса в качестве кандидата по методике[1], ведь для получения значения пороговой функции равного 1.0 обязательным условием является тот факт, что у класса-кандидата имеются все поля и методы из сигнатуры утиного поля.

Результаты для различных значений порога

Результаты моделирования различных пороговых значений представлены в таблице 1. Результаты для обеих функций приведены в одной таблице и разделены знаком «слэш». В первом столбце представлены различные значения порога, в остальных результаты по проектам. Сам результат представляет из себя долю корректных срабатываний алгоритма типизации, аналогичную примененной при оценке результатов в [1]. Данные результаты также представлены на графиках, показанных на рис. 1.

Важным результатом является то, что методики capacity и frequency показывают практически идентичные результаты, вернее, capacity дает даже чуть лучшие результаты. Поэтому, учитывая большуя сложность вычисления frequency (необ-

ходим сбор и подсчет данных о частоте обращения к полям и методам утиного поля), более эффективно использовать функцию capacity. Другие выводы будут приведены ниже после представления дополнительных результатов.

Таблица 1. Результаты динамической проверки типизации с использованием функций capacity/ frequency.

Порог	Logilab	Pylint	Bazaar
0,10	0,68/0,68	0,50/0,50	0,63/0,63
0,15	0,68/0,68	0,50/0,50	0,62/0,63
0,20	0,68/0,68	0,50/0,50	0,62/0,61
0,25	0,68/0,68	0,50/0,50	0,62/0,61
0,30	0,68/0,68	0,50/0,50	0,61/0,60
0,35	0,68/0,68	0,50/0,50	0,60/0,60
0,40	0,68/0,68	0,50/0,50	0,60/0,60
0,45	0,68/0,68	0,50/0,50	0,60/0,60
0,50	0,68/0,68	0,50/0,50	0,60/0,60
0,55	0,68/0,68	0,20/0,20	0,58/0,57
0,60	0,68/0,68	0,20/0,20	0,58/0,57
0,65	0,68/0,68	0,20/0,20	0,57/0,57
0,70	0,68/0,68	0,20/0,20	0,53/0,55
0,75	0,68/0,68	0,20/0,20	0,50/0,51
0,80	0,68/0,68	0,20/0,20	0,50/0,50
0,85	0,68/0,68	0,20/0,20	0,50/0,50
0,90	0,68/0,68	0,20/0,20	0,49/0,50
0,95	0,68/0,68	0,20/0,20	0,49/0,50

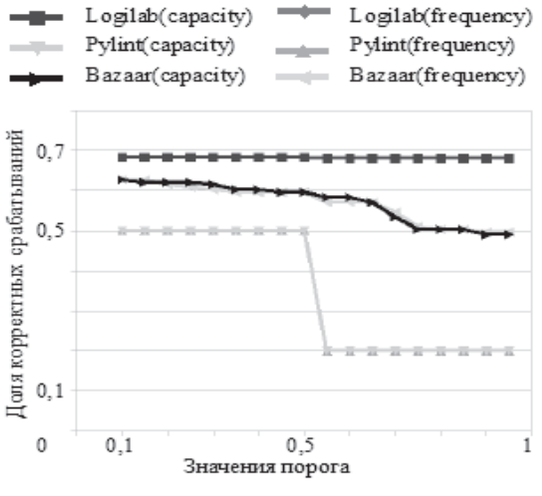

Рис. 1. График зависимости результатов динамической проверки от порога

Дополнительные результаты

Помимо динамического анализа, как и в оригинальной методике, результаты также можно оценить сразу после получения *UCR* с типами полей классов, установленными по предложенной методике. Эти первичные результаты также представляют большой интерес.

1. Множество утиных полей проекта $DUCK\_FIELDS$. Отметим, что каждое утиное поле должно рассматриваться в связи с классом, которому оно принадлежит, но для упрощения изложения, положим, что поля с одинаковыми именами из разных классов представлены различными элементами множества. Множество уникальных идентификаторов классов проекта UCR_ID. Определим отношение $FIELD\_TYPES \subseteq DUCK\_FIELDS \times UCR\_ID$, содержащее пары утиных полей и идентификаторов тех классов, которые были подобраны в качестве кандидатов для данного поля:

$$FIELD\_TYPES = \left\{ \begin{array}{l} (d_f, c) | \\ d_f \in DUCK\_FIELDS \\ c \in UCR\_ID \end{array} \right\}.$$

Тогда область определения данного бинарного отношения содержит те утиные поля, для которых был подобран хотя бы один класс-кандидат. Используя метрику:

$$typed\_fields = \frac{|DomFIELD\_TYPES|}{|DUCK\_FIELDS|}, \quad (3)$$

можно оценить эффективность методики типизации, основываясь на доле тех утиных полей, для которых был подобран хотя бы один класс-кандидат.

2. Интерес представляет и область значений данного отношения, она содержит те классы, которые были использованы в качестве классов-кандидатов хотя бы один раз.

$$cand\_classes = \frac{|\mathrm{Im}FIELD\_TYPES|}{|UCR\_ID|}. \quad (4)$$

Метрика из формулы (4) позволяет оценить долю таких классов. Приведем полученные результаты. По указанным ранее причинам результаты получались только с использованием пороговой функции capacity.

В таблице 2 представлены результаты расчета метрики $typed\_fields$ – см. формулу (3), на рис. 2 представлен соответствующий график. Ниже представлены результаты расчета метрики

cand _ classes – см. формулу (4) и таблицу 3, а также график зависимости на рис. 3.

Заключение

Разработана и успешно применена методика подбора классов-кандидатов для типов полей классов.

Таблица 2. Доля утиных полей с подобранными кандидатами на разных порогах.

Порог	Logilab	Pylint	Bazaar
0,05	0,83	0,70	0,94
0,10	0,83	0,70	0,94
0,15	0,83	0,67	0,94
0,20	0,83	0,66	0,94
0,25	0,83	0,66	0,94
0,30	0,82	0,65	0,93
0,35	0,82	0,65	0,92
0,40	0,82	0,65	0,92
0,45	0,81	0,65	0,92
0,50	0,81	0,65	0,92
0,55	0,80	0,59	0,88
0,60	0,79	0,59	0,87
0,65	0,79	0,59	0,87
0,70	0,78	0,59	0,85
0,75	0,77	0,57	0,84
0,80	0,77	0,57	0,83
0,85	0,77	0,57	0,82
0,90	0,77	0,56	0,81
0,95	0,77	0,56	0,80

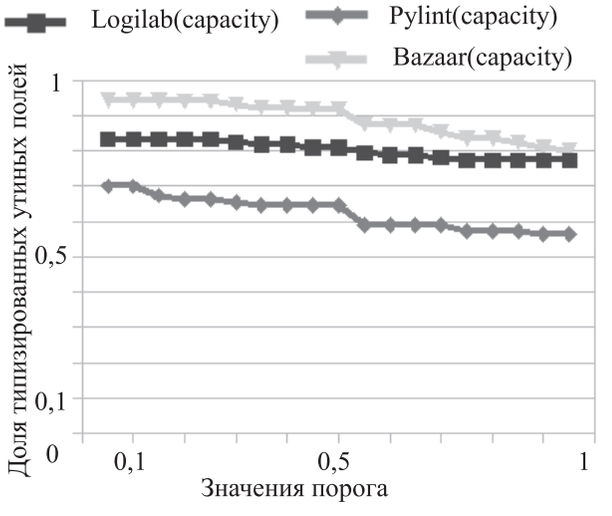

Рис. 2. Зависимость доли типизированных утиных полей от порога

На основе исследования нескольких крупных Python-проектов были исследованы результаты для разных порогов, что позволило определить оптимальные значения порога для использования.

Таблица 3. Доля классов, подобранных в качестве кандидатов на разных порогах.

Порог	Logilab	Pylint	Bazaar
0,05	0,92	0,69	0,77
0,10	0,92	0,69	0,77
0,15	0,90	0,69	0,76
0,20	0,90	0,69	0,76
0,25	0,90	0,69	0,76
0,30	0,90	0,68	0,76
0,35	0,89	0,68	0,75
0,40	0,89	0,68	0,75
0,45	0,89	0,68	0,75
0,50	0,89	0,68	0,75
0,55	0,88	0,65	0,71
0,60	0,88	0,65	0,71
0,65	0,88	0,65	0,71
0,70	0,88	0,65	0,71
0,75	0,87	0,65	0,71
0,80	0,87	0,65	0,71
0,85	0,87	0,65	0,71
0,90	0,87	0,65	0,70
0,95	0,87	0,65	0,70

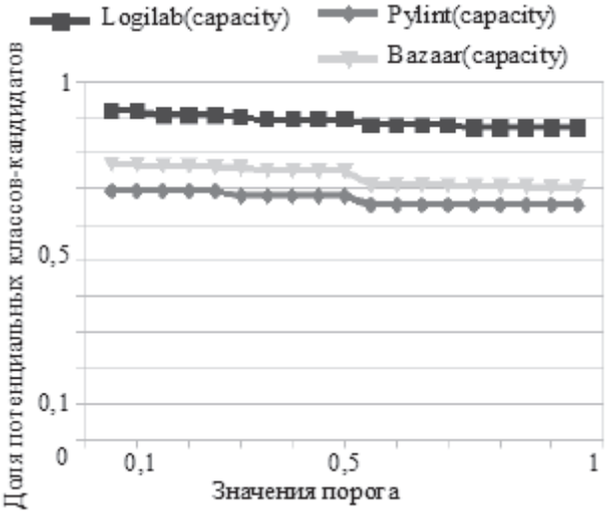

Рис. 3. Зависимость доли используемых классов от порога

Значительные изменения результатов срабатывания происходят на участке значений порога от 0,5 до 0,7. Значения порога менее 0,5 брать не имеет смысла, так как для двух проектов это вообще никак не отражается на результатах. При этом могут появиться ложные срабатывания. На значениях порога выше 0,7 результаты довольно близки к оригинальной методике, требовавшей полного соответствия сигнатуры класса-кандидата и сигнатуры утиного поля.

После успешного применения методик типизации для универсального классового представления была выдвинута идея о применении данного подхода и в генераторе универсального представления потока управления (UCFR), прототип которого представлен в [5]. Для Python в этом представлении данный подход может быть применен для выполнения типизации локальных имен функциональных блоков (локальных переменных и аргументов функций).

Литература

1. Зубов М.В., Пустыгин А.Н., Старцев Е.В. Получение типов данных в языках с динамической типизацией для статического анализа исходного кода с помощью универсального классового представления // Вестник Астрах. ГУ. Серия: Управление, вычислительная техника и информатика. №2, 2013. – С. 66-74.
2. Index (Logilab.org) / http://www.logilab.org (15.06.2014).
3. Pylint-code analysis for Python / www.pylint.org / http://www.pylint.org/ (15.06.2014).
4. Bazaar / http://bazaar.canonical.com/en/ (15.06.2014)
5. Зубов М.В., Пустыгин А.Н., Старцев Е.В. Построение универсального представления графа потока управления для статического анализа исходного кода // Тезисы докладов IX НК «СПО в высшей школе». М.: Альт Линукс, 2014. – С. 46-51.

DATA TYPES EXTRACTION USING THRESHOLD FUNCTIONS FOR INTERMEDIATE REPRESENTATION CREATION OF SOURCE CODE WRITTEN IN DYNAMIC-TYPE LANGUAGES

Zubov M.V., Pustygin A.N., Startsev E.V.

This article shows approach for getting information of data types in languages with dynamic typing. It uses threshold functions to find right one. These functions are described in math model, which shows the whole approach. Results of its application, which was obtained for several projects, show efficiency of using it.

Keywords: *static analysis, intermediate representation, source code, dynamic typing, math modeling, Python.*

Зубов Максим Валерьевич, аспирант Кафедры компьютерной безопасности и прикладной алгебры (КБ и ПА) Челябинского государственного университета (ЧелГУ). Тел. 8-961-784-45-31. E-mail: zubovmv@gmail.com

Пустыгин Алексей Николаевич, к.т.н., доцент Кафедры КБ и ПА ЧелГУ. Тел. 8-905-835-98-68. E-mail: p2008an@rambler.ru

Старцев Евгений Владимирович, аспирант Кафедры КБ и ПА ЧелГУ. Тел. 8-961-789-69-23. E-mail: slayer-gurgen@yandex.ru

НОВЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

УДК 65.011.56

ИНТЕГРИРОВАННЫЕ СИСТЕМЫ УПРАВЛЕНИЯ ПРОМЫШЛЕННЫМ ПРЕДПРИЯТИЕМ: МЕТОДОЛОГИЯ СОЗДАНИЯ

Матвеева Е.А.

В статье рассматриваются методологические вопросы создания интегрированной системы управления промышленным предприятием, позволяющей реализовать задачи управления производственной деятельностью, позволяющие получить экономический эффект.

Ключевые слова: организация, управление, промышленное предприятие, методология создания, информатизация и компьютеризация, экономическая эффективность.