Верификация асимптотической оценки временной сложности в задачах динамического программирования

Григорянц Сергей Арменович

Московский физико-технический институт
Физтех-школа Прикладной Математики и Информатики
Кафедра дискретной математики

Научный руководитель: Дашков Евгений Владимирович

28 июня 2021 г.

Постановка задачи

Цель работы

Постановка задачи

Цель работы

- Исследование методов формальной верификации корректности и асимптотики алгоритмов, предложенных в статье Шагеро и др. [3].

Постановка задачи

Цель работы

- Исследование методов формальной верификации корректности и асимптотики алгоритмов, предложенных в статье Шагеро и др. [3].
- Применение исследованных методов на примере верификации алгоритма динамического программирования LCS.

Зачем нужна формальная верификация?

Сферы применения

Зачем нужна формальная верификация?

Сферы применения

- Аппаратное обеспечение – Intel [4].

Зачем нужна формальная верификация?

Сферы применения

- Аппаратное обеспечение – Intel [4].
- Криптография – Scilla [9], CertiK [10].

Зачем нужна формальная верификация?

Сферы применения

- Аппаратное обеспечение – Intel [4].
- Криптография – Scilla [9], CertiK [10].
- Критическое ПО – CompCert [7], seL4 [6].

Зачем нужна формальная верификация?

Сферы применения

- Аппаратное обеспечение – Intel [4].
- Криптография – Scilla [9], CertiK [10].
- Критическое ПО – CompCert [7], seL4 [6].
- Медицина, банковское дело, транспортные технологии, и т.д.

Что бывает, если не верифицировать ПО

Истории неудач

Что бывает, если не верифицировать ПО

Истории неудач

- Сорвалась миссия НАСА Mars Climate Orbiter.

Что бывает, если не верифицировать ПО

Истории неудач

- Сорвалась миссия НАСА Mars Climate Orbiter.
- Ненадлежащее тестирование Лондонской службы скорой помощи привело к гибели людей.

Что бывает, если не верифицировать ПО

Истории неудач

- Сорвалась миссия НАСА Mars Climate Orbiter.
- Ненадлежащее тестирование Лондонской службы скорой помощи привело к гибели людей.
- Самолет Airbus A320 разбился на демонстрационном полете из-за ошибке в софте.

Что бывает, если не верифицировать ПО

Истории неудач

- Сорвалась миссия НАСА Mars Climate Orbiter.
- Ненадлежащее тестирование Лондонской службы скорой помощи привело к гибели людей.
- Самолет Airbus A320 разбился на демонстрационном полете из-за ошибке в софте.
- Много страшных историй: [2]

Coq

Coq

- Coq – программное средство доказательства теорем.

Coq

- Coq – программное средство доказательства теорем.
- Coq основан на теории типов (Исчисление Индуктивных Конструкций, Calculus of Inductive Constructions, CIC)

Coq

- Coq – программное средство доказательства теорем.
- Coq основан на теории типов (Исчисление Индуктивных Конструкций, Calculus of Inductive Constructions, CIC)
- CIC способна представлять:

Coq

- Coq – программное средство доказательства теорем.
- Coq основан на теории типов (Исчисление Индуктивных Конструкций, Calculus of Inductive Constructions, CIC)
- CIC способна представлять:
  - Функциональные программы в стиле ML.

Coq

- Coq – программное средство доказательства теорем.
- Coq основан на теории типов (Исчисление Индуктивных Конструкций, Calculus of Inductive Constructions, CIC)
- CIC способна представлять:
  - Функциональные программы в стиле ML.
  - Логические утверждения и их формальные доказательства.

Coq

- Coq – программное средство доказательства теорем.
- Coq основан на теории типов (Исчисление Индуктивных Конструкций, Calculus of Inductive Constructions, CIC)
- CIC способна представлять:
  - Функциональные программы в стиле ML.
  - Логические утверждения и их формальные доказательства.
- Утверждения и доказательства представляются с помощью Соответствия Карри — Ховарда:

Coq

- Coq – программное средство доказательства теорем.
- Coq основан на теории типов (Исчисление Индуктивных Конструкций, Calculus of Inductive Constructions, CIC)
- CIC способна представлять:
  - Функциональные программы в стиле ML.
  - Логические утверждения и их формальные доказательства.
- Утверждения и доказательства представляются с помощью Соответствия Карри — Ховарда:
  - Пропозициональное утверждение $\Longleftrightarrow$ Тип.

Coq

- Coq – программное средство доказательства теорем.
- Coq основан на теории типов (Исчисление Индуктивных Конструкций, Calculus of Inductive Constructions, CIC)
- CIC способна представлять:
  - Функциональные программы в стиле ML.
  - Логические утверждения и их формальные доказательства.
- Утверждения и доказательства представляются с помощью Соответствия Карри — Ховарда:
  - Пропозициональное утверждение $\iff$ Тип.
  - Доказательство утверждения $\iff$ Элемент(терм) данного типа.

Coq

- Coq – программное средство доказательства теорем.
- Coq основан на теории типов (Исчисление Индуктивных Конструкций, Calculus of Inductive Constructions, CIC)
- CIC способна представлять:
  - Функциональные программы в стиле ML.
  - Логические утверждения и их формальные доказательства.
- Утверждения и доказательства представляются с помощью Соответствия Карри — Ховарда:
  - Пропозициональное утверждение $\iff$ Тип.
  - Доказательство утверждения $\iff$ Элемент(терм) данного типа.
- Vernacular – язык команд Coq.

Логика Хоара

Логика Хоара

- Логика Хоара [5] – формальная система, которая позволяет рассуждать о корректности императивных программ.

Логика Хоара

- Логика Хоара [5] – формальная система, которая позволяет рассуждать о корректности императивных программ.
- Тройка Хоара $\{P\}C\{Q\}$

Логика Хоара

- Логика Хоара [5] – формальная система, которая позволяет рассуждать о корректности императивных программ.
- Тройка Хоара $\{P\}C\{Q\}$
  - P и Q – предикаты на множестве состояний.

Логика Хоара

- Логика Хоара [5] – формальная система, которая позволяет рассуждать о корректности императивных программ.
- Тройка Хоара $\{P\}C\{Q\}$
  - P и Q – предикаты на множестве состояний.
  - C – некоторая последовательность команд.

Логика Хоара

- Логика Хоара [5] – формальная система, которая позволяет рассуждать о корректности императивных программ.
- Тройка Хоара $\{P\}C\{Q\}$
  - P и Q – предикаты на множестве состояний.
  - C – некоторая последовательность команд.
- Тройка выводится $\iff$ если текущее состояние удовлетворяет утверждению P, то после выполнения на нем программы C она будет удовлетворять утверждению Q.

Логика Хоара

- Логика Хоара [5] – формальная система, которая позволяет рассуждать о корректности императивных программ.
- Тройка Хоара $\{P\}C\{Q\}$
  - P и Q – предикаты на множестве состояний.
  - C – некоторая последовательность команд.
- Тройка выводится $\iff$ если текущее состояние удовлетворяет утверждению P, то после выполнения на нем программы C она будет удовлетворять утверждению Q.
- Пример: $\{x = 3 \wedge y = 4\}[x := x + y]\{x = 7 \wedge y = 4\}$.

Логика Хоара

- Логика Хоара [5] – формальная система, которая позволяет рассуждать о корректности императивных программ.
- Тройка Хоара $\{P\}C\{Q\}$
  - P и Q – предикаты на множестве состояний.
  - C – некоторая последовательность команд.
- Тройка выводится $\iff$ если текущее состояние удовлетворяет утверждению P, то после выполнения на нем программы C она будет удовлетворять утверждению Q.
- Пример: $\{x = 3 \wedge y = 4\}[x := x + y]\{x = 7 \wedge y = 4\}$.
- Логика Хоара неприменима, если в программе есть общее состояние(указатели).

Логика Хоара

- Логика Хоара [5] – формальная система, которая позволяет рассуждать о корректности императивных программ.
- Тройка Хоара $\{P\}C\{Q\}$
  - P и Q – предикаты на множестве состояний.
  - C – некоторая последовательность команд.
- Тройка выводится $\iff$ если текущее состояние удовлетворяет утверждению P, то после выполнения на нем программы C она будет удовлетворять утверждению Q.
- Пример: $\{x = 3 \wedge y = 4\}[x := x + y]\{x = 7 \wedge y = 4\}$.
- Логика Хоара неприменима, если в программе есть общее состояние(указатели).
- Пример $\{x \mapsto 3 \wedge y \mapsto 3\}[*x := 4]\{x \mapsto 4 \wedge y \mapsto ?\}$.

Сепарационная логика

Сепарационная логика

- Сепарационная Логика [8] – расширение логики Хоара.

Сепарационная логика

- Сепарационная Логика [8] – расширение логики Хоара.
- Вводится дополнительна операция $P * Q$ – разделяющая конъюнкция.

Сепарационная логика

- Сепарационная Логика [8] – расширение логики Хоара.
- Вводится дополнительна операция $P * Q$ – разделяющая конъюнкция.
  - Текущее состояние может быть разделено на два непересекающихся в адресном пространстве состояния.

Сепарационная логика

- Сепарационная Логика [8] – расширение логики Хоара.
- Вводится дополнительна операция $P * Q$ – разделяющая конъюнкция.
  - Текущее состояние может быть разделено на два непересекающихся в адресном пространстве состояния.
  - Каждое из них удовлетворяет P и Q соответственно.

Сепарационная логика

- Сепарационная Логика [8] – расширение логики Хоара.
- Вводится дополнительна операция $P * Q$ – разделяющая конъюнкция.
  - Текущее состояние может быть разделено на два непересекающихся в адресном пространстве состояния.
  - Каждое из них удовлетворяет P и Q соответственно.
- Новое правило вывода(правило разделения):

$$\frac{\{P\}C\{Q\}}{\{P * R\}C\{Q * R\}}$$

Сепарационная логика

- Сепарационная Логика [8] – расширение логики Хоара.
- Вводится дополнительна операция $P * Q$ – разделяющая конъюнкция.
  - Текущее состояние может быть разделено на два непересекающихся в адресном пространстве состояния.
  - Каждое из них удовлетворяет P и Q соответственно.
- Новое правило вывода(правило разделения):

$$\frac{\{P\}C\{Q\}}{\{P * R\}C\{Q * R\}}$$

- Пример: $\{x \mapsto 3 * y \mapsto 3\}[*x = 4]\{x \mapsto 4 * y \mapsto 3\}$.

Сепарационная Логика с временными ресурсами

Сепарационная Логика с временными ресурсами

- Основная идея [1] – введение времени как потребляемого ресурса внутрь предиката состояния.

Сепарационная Логика с временными ресурсами

- Основная идея [1] – введение времени как потребляемого ресурса внутрь предиката состояния.
- Предикат $1 означает возможность сделать один шаг вычисления.

Сепарационная Логика с временными ресурсами

- Основная идея [1] – введение времени как потребляемого ресурса внутрь предиката состояния.
- Предикат \$1 означает возможность сделать один шаг вычисления.
- Предикат \$$n$ – разделяющая конъюнкция n предикатов \$1.

Сепарационная Логика с временными ресурсами

- Основная идея [1] – введение времени как потребляемого ресурса внутрь предиката состояния.
- Предикат \$1 означает возможность сделать один шаг вычисления.
- Предикат \$$n$ – разделяющая конъюнкция n предикатов \$1.
- Ресурс поглощается при выполнении шага вычисления.

Сепарационная Логика с временными ресурсами

- Основная идея [1] – введение времени как потребляемого ресурса внутрь предиката состояния.
- Предикат \$1 означает возможность сделать один шаг вычисления.
- Предикат \$$n$ – разделяющая конъюнкция n предикатов \$1.
- Ресурс поглощается при выполнении шага вычисления.
- Пример: $\{\$10 * x \mapsto 3 * y \mapsto 3\}[*x = 4]\{\$9 * x \mapsto 4 * y \mapsto 3\}$.

Реализация LCS

```
let lcs (a : int array) (b : int array) : int array =
  let n = Array.length a in
  let m = Array.length b in
  let c = Array.make ((n+1)*(m+1)) [] in
  for i = 1 to n do
    for j = 1 to m do
      if a.(i-1) = b.(j-1)
      then c.(i*(m+1) + j) <-
             List.append c.((i-1)*(m+1) + j - 1) [a.(i-1)]
      else if List.length c.((i-1)*(m+1) + j) >
             List.length c.(i*(m+1) + j - 1)
        then c.(i*(m+1) + j) <- c.((i-1)*(m+1) + j)
        else c.(i*(m+1) + j) <- c.(i*(m+1) + j - 1)
      done
    done;
  Array.of_list c.((n+1)*(m+1)-1);;
```

Определение SubSeq

```
Inductive SubSeq {A:Type} : list A -> list A -> Prop :=
 | SubNil (l:list A) : SubSeq nil l
 | SubCons1 (x:A) (l1 l2:list A) (H: SubSeq l1 l2) :
                  SubSeq l1 (x::l2)
 | SubCons2 (x:A) (l1 l2:list A) (H: SubSeq l1 l2) :
                  SubSeq (x::l1) (x::l2).
```

Определение Lcs

```
Definition Lcs {A: Type} l l1 l2 :=
  SubSeq l l1 /\ SubSeq l l2 /\
  (forall l': list A, SubSeq l' l1 /\ SubSeq l' l2 ->
          length l' <= length l).
```

Необходимые свойства Lcs

```
Lemma lcs_app_eq: forall (l1 l2 l: list int) (x: int),
  Lcs l l1 l2 -> Lcs (l & x) (l1 & x) (l2 & x).

Lemma lcs_app_neq: forall (l1 l2 l l': list int) (x y : int),
  x <> y -> Lcs l (l1&x) l2 -> Lcs l' l1 (l2&y) ->
          length l' <= length l -> Lcs l (l1&x) (l2&y).
```

Формулировка основной теоремы

```
Lemma lcs_spec:
  spec0
    (product_filterType Z_filterType Z_filterType)
    ZZle
  ( fun cost =>
  forall (l1 l2 : list int) p1 p2,
  app lcs [p1 p2]
  PRE (\$(cost (LibListZ.length l1, LibListZ.length l2)) \*
  p1 ~> Array l1 \* p2 ~> Array l2)
  POST (fun p => Hexists (l : list int), p ~> Array l \*
          \[Lcs l l1 l2]))
  (fun '(n,m) => n * m).
```

Инвариант цикла

```
xfor_inv (fun (i:int) =>
Hexists (x' : list (list int)),
p1 ~> Array l1 \*
p2 ~> Array l2 \*
c ~> Array x' \*
\[length x' = (n+1)*(m+1)] \*
\[forall i1 i2 : int, 0 <= i1 < i -> 0 <= i2 <= m ->
Lcs x'[i1*(m+1) + i2] (take i1 l1 ) (take i2 l2) ] \*
\[forall i', i*(m+1) <= i' < (n+1)*(m+1) ->
x'[i'] = nil ]).
```

Возможные темы дальнейшего исследования

Возможные темы дальнейшего исследования

- Реализация функции HForall.

Возможные темы дальнейшего исследования

- Реализация функции HForall.
  - Реализовать саму функцию.

Возможные темы дальнейшего исследования

- Реализация функции HForall.
  - Реализовать саму функцию.
  - Доказать ее свойства для интеграции с тактикой hsimpl.

Возможные темы дальнейшего исследования

- Реализация функции HForall.
  - Реализовать саму функцию.
  - Доказать ее свойства для интеграции с тактикой hsimpl.
- Формальная верификация асимптотики вероятностных алгоритмов.

Список литературы I

Robert Atkey. “Amortised Resource Analysis with Separation Logic”. в: *Programming Languages and Systems*. под ред. Andrew D. Gordon. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, с. 85—103. ISBN: 978-3-642-11957-6.

N. Dershowitz. *SOFTWARE HORROR STORIES*. URL: https://www.cs.tau.ac.il/~nachumd/verify/horror.html.

Armaël Guéneau, Arthur Charguéraud и François Pottier. “A Fistful of Dollars: Formalizing Asymptotic Complexity Claims via Deductive Program Verification”. в: *Programming Languages and Systems*. под ред. Amal Ahmed. Cham: Springer International Publishing, 2018, с. 533—560. ISBN: 978-3-319-89884-1.

J. Harrison. “Formal verification at Intel”. в: *18th Annual IEEE Symposium of Logic in Computer Science, 2003. Proceedings.* 2003, с. 45—54. DOI: 10.1109/LICS.2003.1210044.

Список литературы II

C. A. R. Hoare. “An Axiomatic Basis for Computer Programming”. в: *Commun. ACM* 12.10 (окт. 1969), с. 576—580. ISSN: 0001-0782. DOI: 10.1145/363235.363259. URL: https://doi.org/10.1145/363235.363259.

Gerwin Klein и др. “SeL4: Formal Verification of an OS Kernel”. в: *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*. SOSP ’09. Big Sky, Montana, USA: Association for Computing Machinery, 2009, с. 207—220. ISBN: 9781605587523. DOI: 10.1145/1629575.1629596. URL: https://doi.org/10.1145/1629575.1629596.

Xavier Leroy. “Formal Verification of a Realistic Compiler”. в: *Commun. ACM* 52.7 (июль 2009), с. 107—115. ISSN: 0001-0782. DOI: 10.1145/1538788.1538814. URL: https://doi.org/10.1145/1538788.1538814.

Список литературы III

J.C. Reynolds. “Separation logic: a logic for shared mutable data structures”. в: *Proceedings 17th Annual IEEE Symposium on Logic in Computer Science*. 2002, с. 55—74. DOI: 10.1109/LICS.2002.1029817.

Ilya Sergey и др. “Safer Smart Contract Programming with Scilla”. в: *Proc. ACM Program. Lang.* 3.OOPSLA (окт. 2019). DOI: 10.1145/3360611. URL: https://doi.org/10.1145/3360611.

CertiK team. *CertiK framework*. URL: https://www.certik.io/.