多重配列を扱う命令型プログラムの所有権と篩型を組み合わせによる形式検証

藤原　佑輔・松下　祐介・末永　幸平・五十嵐　淳
京都大学情報学研究科

目標: 命令型言語で多重配列の検証を行う

背景: ポインタ演算を含むConSORT [Toman+, '20] [Tanaka+, '24]

エイリアシングと破壊的代入を含むプログラムの検証器

整数配列と ポインタ演算とを含む低レベルプログラムの静的検証器

篩型 $\{v{:}\mathrm{int} \mid \varphi\}$

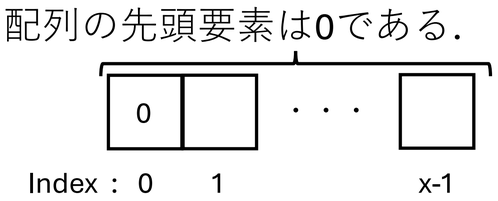
述語φで制約付けされた整数型

参照型 $(\lambda i.\ \tau)\ \mathbf{ref}^{r}$

所有権関数

配列へのポインタ型
i番目の要素の型はτ(iに依存可)

すべての要素は読み書き可能で,
配列の先頭要素は0である.



型付け例

整数配列の初期化をするプログラム

```
1  //init:⟨x : int, p : (λi.{ν : int | ⊤}) ref^[0,x−1]↦1⟩
   → ⟨x : int, p : (λi.{ν : int | ν = 0}) ref^[0,x−1]↦1 | int⟩
2  init (x, p){
3    if x = 0 then { 0 }
4    else {
5      p := 0;
6      //p:(λi.{ν : int | i = 0 ⟹ ν = 0}) ref^[0,x−1]↦1
7      let q = p + 1 in
8      //p:(λi.{ν : int | ν = 0}) ref^[0,0]↦1
9        q:(λi.{ν : int | ⊤}) ref^[0,x−2]↦1
10     init(x-1, q);
11     //p:(λi.{ν : int | ν = 0}) ref^[0,0]↦1
12       q:(λi.{ν : int | ν = 0}) ref^[0,x−2]↦1
13     alias(q = p + 1)
14     //p:(λi.{ν : int | ν = 0}) ref^[0,x−1]↦1
15   }
16 }
```

所有権関数*r*

- 配列のインデックスから各要素の所有権への関数
- 各要素の読み書き権限を表現
- r(i) は0以上1以下の有理数
  - r(i)=1 : i番目の要素が読み書き可能で, 他のエイリアスから読み書き不可
  - 0<r(i)<1:i番目の要素の読み出しが可能
- エイリアス同士の所有権の和は1以下.
- [配列の下限,配列の上限] ↦所有権の値 の形式で表現
- 例： [0,x-1] ↦1 は書き込み可能な長さxの配列

alias式は所有権の再分配や篩型のすり合わせをするためのヒントとして、プログラマに記述される.

既存研究の課題: 型推論アルゴリズムは多重配列を扱っているが、所有権関数にどの変数が依存してよいか不明瞭で、かつ検証器の実装では整数の一次元配列しか扱っていない

アプローチ: *r*の形式化

(所有権項)　$r \quad ::= \quad \varphi \mapsto q, r \mid$

$\mathbf{true} \mapsto q$

- *r*: 所有権項
- φ:整数リテラル、整数変数、配列のインデックスを表す変数が現れる量化子のない一階述語論理の命題
- q: $0 \le q \le 1$を満たす有理数

現在の状況

- プログラムからownership termと型の条件に関わる制約を生成する型推論アルゴリズム を構築
- 下記の例が動く型検証器を実装中
- 健全性の証明は今後の課題

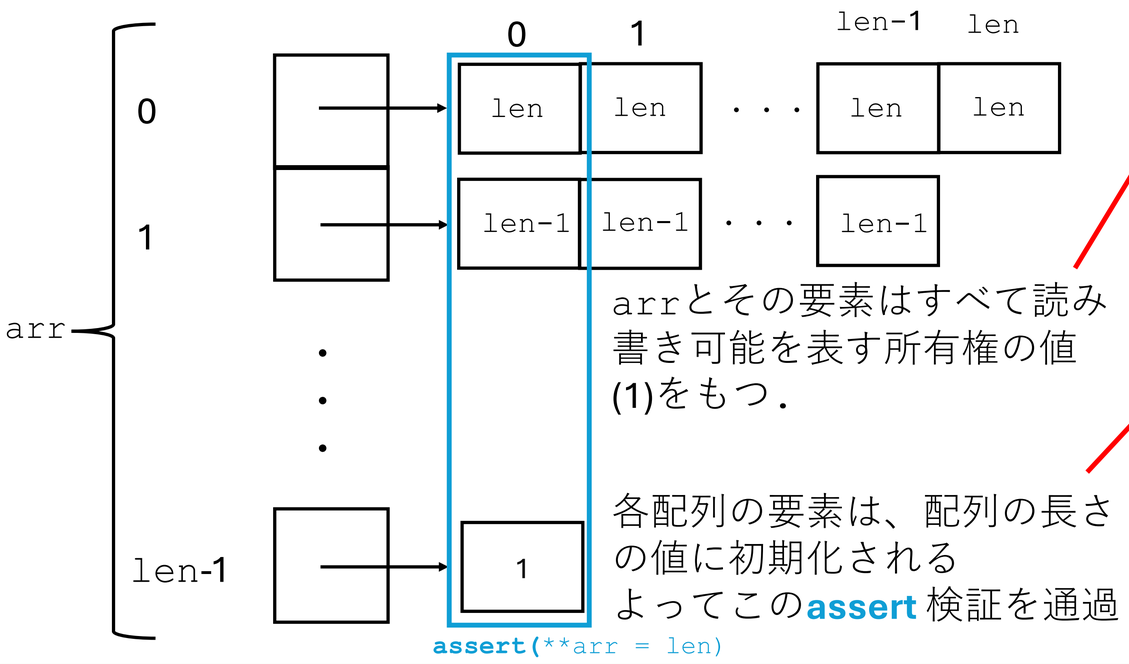
型付けの例

- **int_array_init** ：整数配列arrの先頭からlen個の要素を値numで初期化する関数
- **nested_array_init** ：整数配列の配列arrの先頭からlen個の要素を初期化する関数

```
1  int_array_init (arr, len, num){
2    if len > 0 then {
3      arr := num; let next_arr = arr + 1 in
4      let x = int_array_init(next_arr, len - 1, num) in
5      alias(next_arr = arr + 1)
6    } }
7
8  // nested_array_init:
9  // ⟨arr:(λi1.int ref) ref^(0≤i1∧i1≤len−1↦1), len:int⟩ ->
10 // ⟨arr:(λi1.(λi2.{ν:int | 0 ≤ i2 ∧ i2 ≤ len − i1 − 1
11 //                 ⟹ ν = len − i1}) ref^r2) ref^r1, len : int⟩
12 //  r1 = (0 ≤ i1 ∧ i1 ≤ len − 1 ↦ 1), r2 = (0 ≤ i2 ∧ i2 ≤ len − i1 − 1 ↦ 1)
13 nested_array_init (arr, len){
14   if len > 0 then {
15     let sub_arr = alloc len : int ref in
16     let x = int_array_init(sub_arr, len, len) in
17     arr := sub_arr;
18     // arr:(λi1.(λi2.{ν:int | i1 = 0 ∧ 0 ≤ i2 ∧ i2 ≤ len − 1
19     //      ⟹ ν = len}) ref^r2) ref^r1
20     // r1 = (0 ≤ i1 ∧ i1 ≤ len − 1 ↦ 1)
21     // r2 = (i1 = 0 ∧ 0 ≤ i2 ∧ i2 ≤ len − 1 ↦ 1)
22     assert(**arr = len); let next_arr = arr + 1 in
23     // arr:(λi1.(λi2.{ν:int | i1 = 0 ∧ 0 ≤ i2 ∧ i2 ≤ len − 1
24     //      ⟹ ν = len}) ref^r2) ref^r1
25     // r1 = (i1 = 0 ↦ 1), r2 = (i1 = 0 ∧ 0 ≤ i2 ∧ i2 ≤ len − 1 ↦ 1)
26     // next_arr:(λi1.int ref) ref^(0≤i1∧i1≤len−2↦1)
27     nested_array_init (next_arr, len - 1);
28     // next_arr:(λi1.(λi2.{ν:int | 0 ≤ i2 ∧ i2 ≤ len − i1 − 2
29     //      ⟹ ν = len − i1 − 1}) ref^r2) ref^r1
30     // r1 = (0 ≤ i1 ∧ i1 ≤ len − 2 ↦ 1), r2 = (0 ≤ i2 ∧ i2 ≤ len − i1 − 2 ↦ 1)
31     alias(next_arr = arr + 1);
32     // arr:(λi1.(λi2.{ν:int | 0 ≤ i2 ∧ i2 ≤ len − i1 − 1
33     //      ⟹ ν = len − i1}) ref^r2) ref^r1
34     // r1 = (0 ≤ i1 ∧ i1 ≤ len − 1 ↦ 1), r2 = (0 ≤ i2 ∧ i2 ≤ len − i1 − 1 ↦ 1)
35   } }
```

arr の要素である整数配列はインデックスごとに 異なる長さと値をとる

arr: 整数配列の配列



arrとその要素はすべて読み書き可能を表す所有権の値(1)をもつ.

各配列の要素は、配列の長さの値に初期化される
よってこのassert 検証を通過