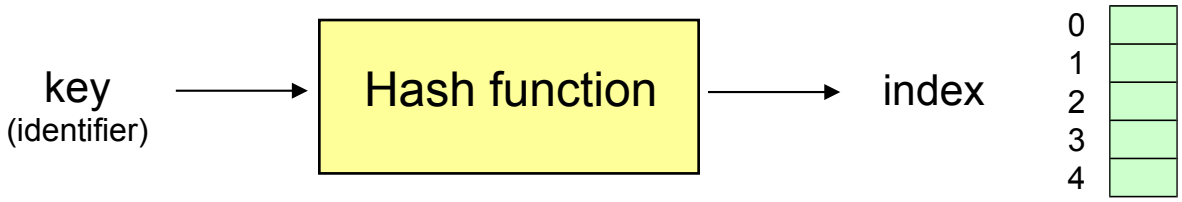
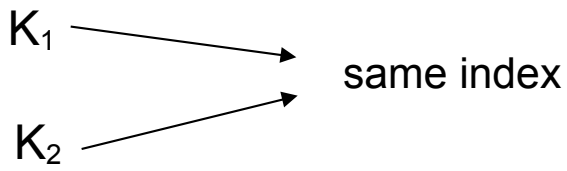
Hash Table

- HT = `array [0..TOT-1] of buckets`



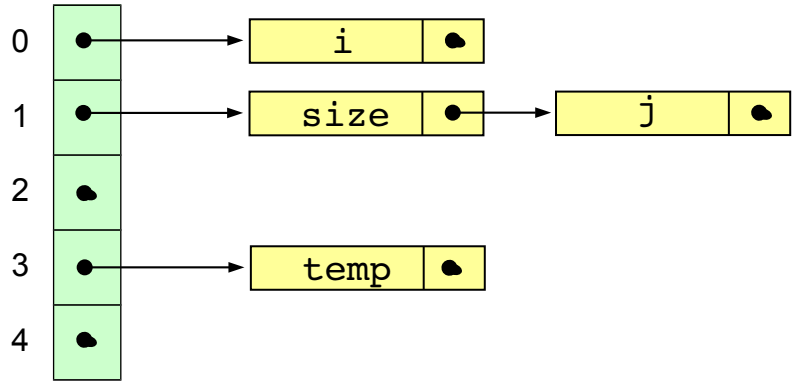
- Problem of collisions:



- Size of array (`TOT`)
  - Normally defined at compiler-construction time (hundreds / thousands)
  - Prime number (better distribution)

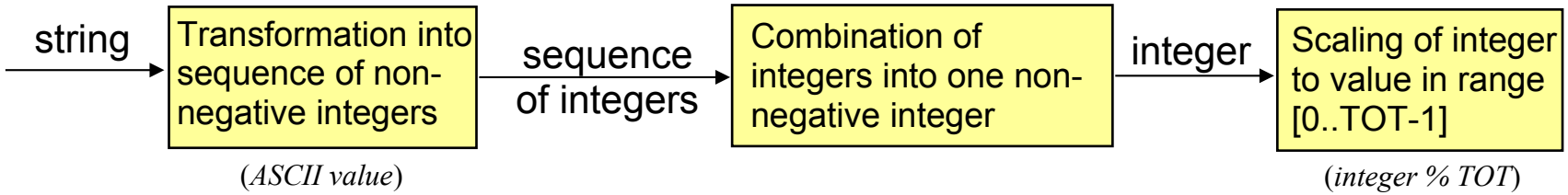
Resolution of Collisions

- Bucket = linear list $\Rightarrow$ collision $\rightarrow$ insert as head of list



Mapping

- Transformation: string of characters → integer ∈ [0..TOT-1] in 3 steps:



- Intermediate step: use of a "weight" α:

$$h = (\alpha^{n-1}c_1 + \alpha^{n-2}c_2 + \ldots + \alpha c_{n-1} + c_n) \% TOT = \left(\sum_{i=1}^{n} \alpha^{n-i}c_i \right) \% TOT$$

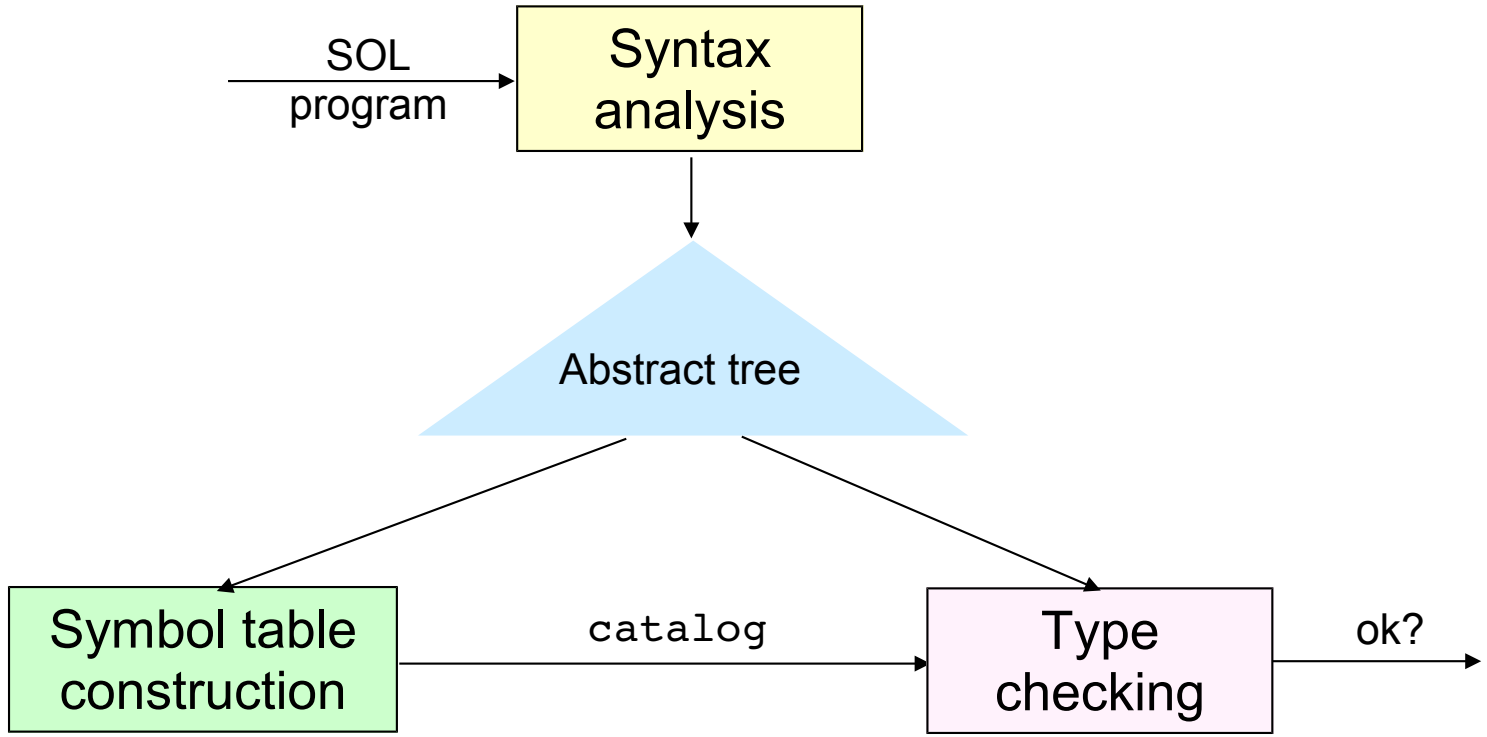
- Pb overflow → insertion of module operation into summation (e.g: α = 16)

```
#define TOT ...
#define SHIFT 4

int hash(char* id)
{  int h=0;

   for(i=0; id[i] != '\0'; i++)
     h = ((h << SHIFT) + id[i]) % TOT;
   return h;
}
```

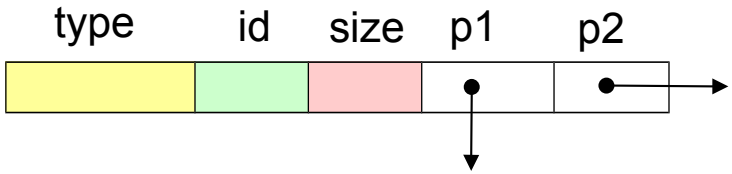
Approach in SOL



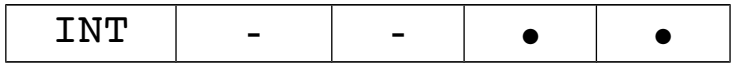
- Symbol table = hierarchical structure isomorphic to environments (static scope)

Schema Representation

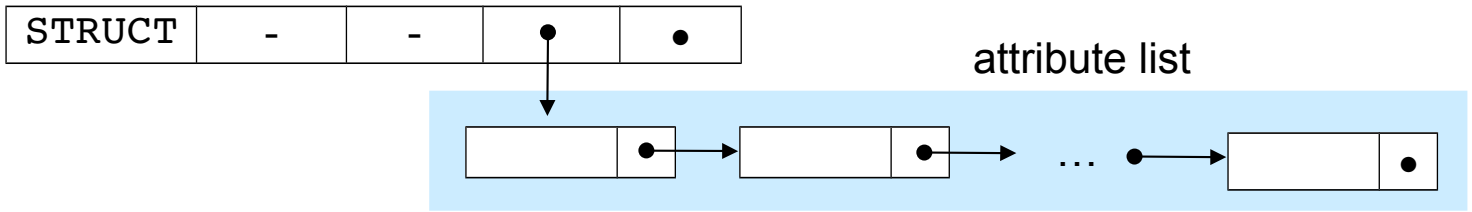
- Node:



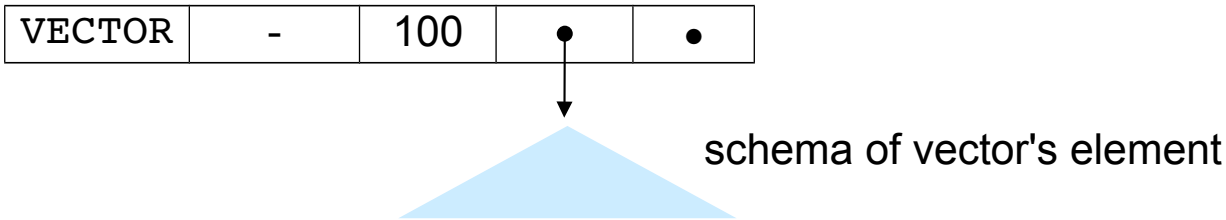
- `type` ∈ { `CHAR`, `INT`, `REAL`, `STRING`, `BOOL`, `STRUCT`, `VECTOR`, `ATTR` }
- `id`: field name
- `size`: array size (`type` = `VECTOR`)
- Atomic type:



- Struct:

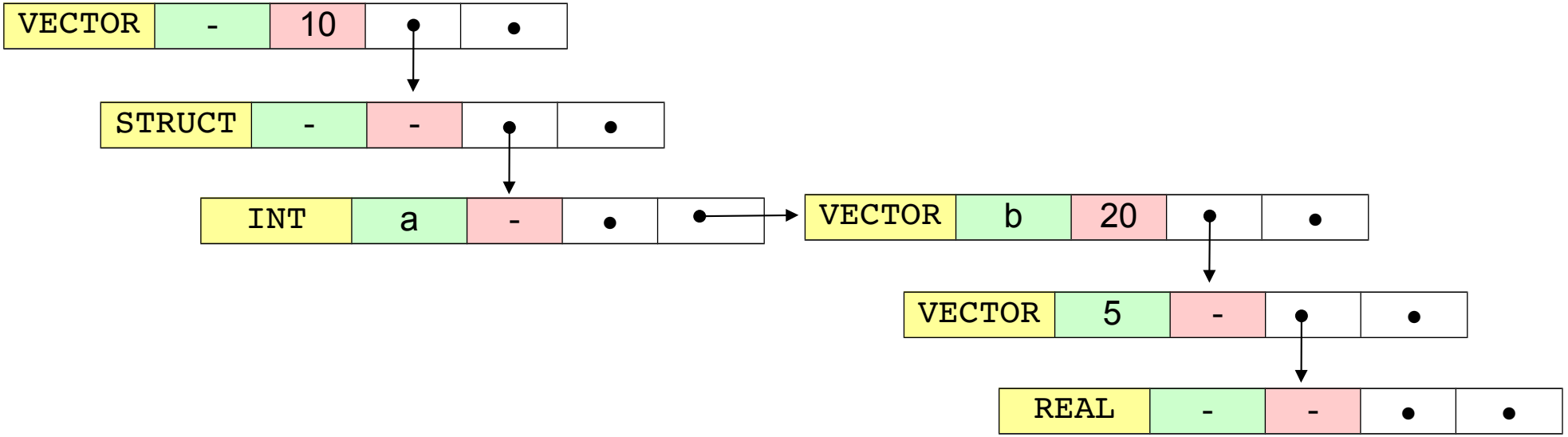


- Vector:

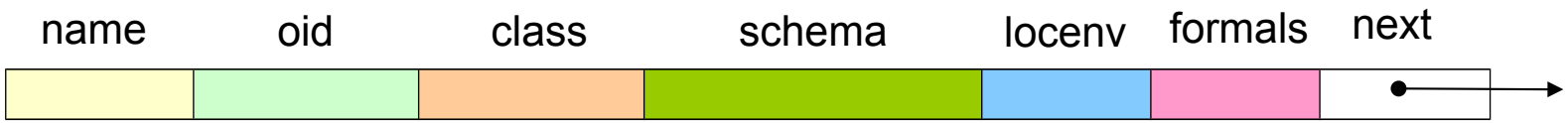


Schema Representation (ii)

```
vector [10] of struct(a: int; b: vector[20] of vector [5] of real;)
```



Symbol Table

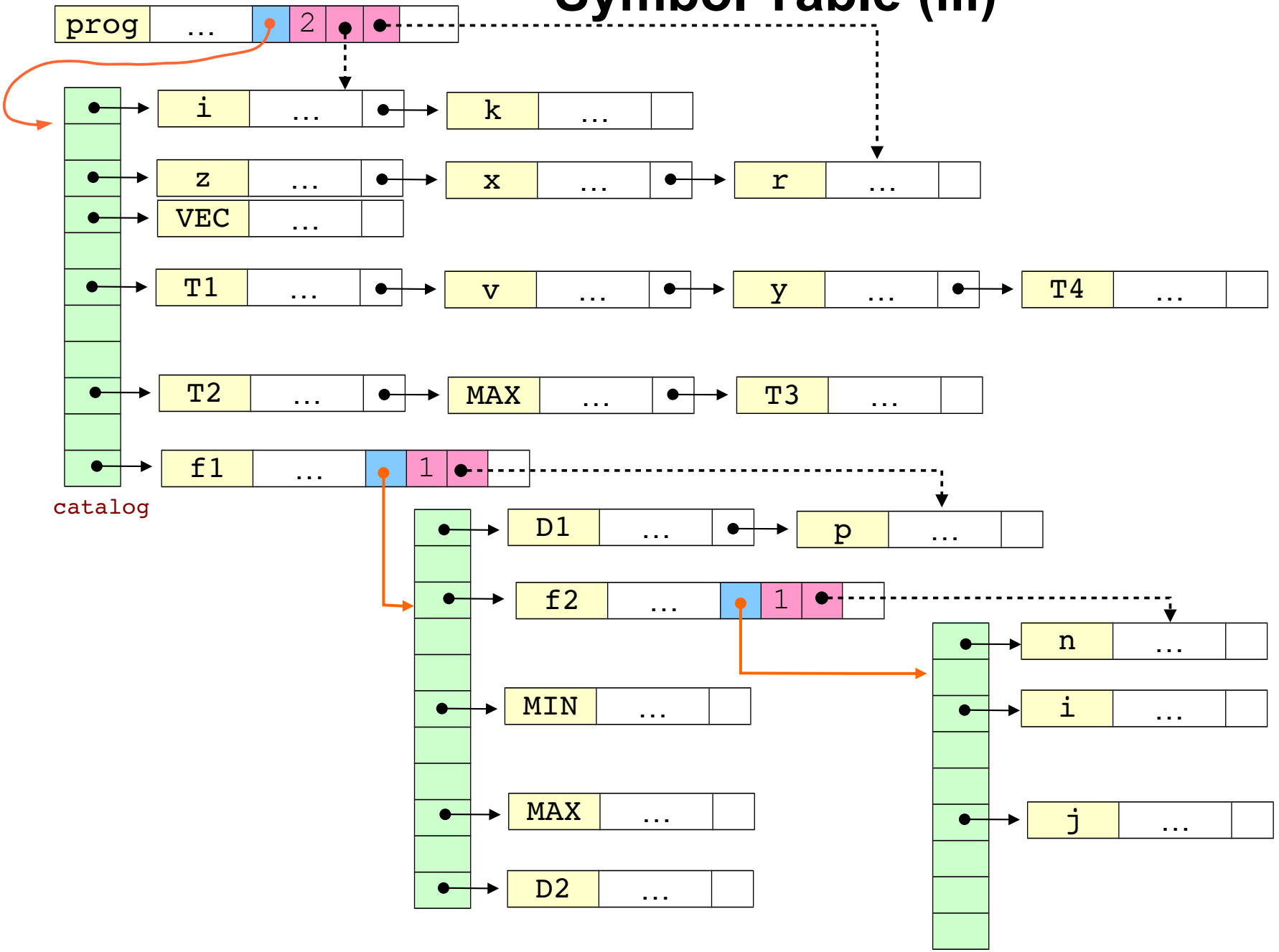


- `name`: pointer to name of identifiers
- `oid`: object identifier
  - par / var / const → `oid` ∈ [1, 2, …] : relative numbering within environment
  - function → `oid` ∈ [1, 2, …] : absolute numbering
- `class` ∈ { `TYPE`, `VAR`, `CONST`, `FUNC`, `PAR` }
- `schema`: pointer to root of schema-tree
- `locenv`: local environment (for functions) = reference to table of local symbols
- `formals`
  - `num` = number of formal parameters
  - `descr` = (dynamic) vector of pointers to symbols in `locenv`
- `next`: pointer to next descriptor (same hash index)

Symbol Table (ii)

```
func prog(i: int; r: struct(a: int; b: string;);): int
  type
      T1, T2: struct(x: real; s: string;);
      T3: T1;
      T4: real;
  var
      x, y, z: T4;
      k: int;
      v: vector [10] of T4;
  const
      MAX: int = i+10;
      VEC: T1 = vector(struct(3.0, "alpha"),struct(4.0, "beta"));
  func f1(p: T2;): T2
    type
      D1: int;
      D2: struct(a: string; b: bool;);
    const
      MIN: int = i – 10;
      MAX: real = x + p;
    ...
    func f2(n: int;): string
      var i, j: int;
      ...
begin prog
   ...
end prog
```

Symbol Table (iii)



Symbol Table (iv)

- Semantic checks during construction of catalog:

  1. No conflict of names within same environment
  2. No conflict of names within same struct schema