# Compiler Practical 2013

## Storage Administration: Implementation

Berthold Hoffmann (B. Gersdorf, T. Röfer)
hof@informatik.uni-bremen.de
Cartesium 2.48

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Universität Bremen

# Structure

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

1. Add Garbage Collection
2. Administration of the Root Set
3. Simple Storage Adjustment
4. Changing the Semantics of Assignments
5. Bonus Tasks

# Adding garbage collection

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

- Changing the administration of the heap
  - Store heap pointer in storage
  - This frees register *R4*
  - *NEW* must set attributes to *NULL*
  - Local variables must be initialized to *NULL*
- Adminitration of the root set
  - One stack contains all references to objects, a second stack contains the others
  - *R4* points to one stack, *R2* points to the other one

# Adding garbage collection(2)

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

- Handle lack of storage
  - *NEW* compares with upper bound of Heap
  - If heap is full, start garbage collection
  - If heap is still full, an error occurs
- Change semantics of assignments
  - Evaluate right-hand side before left-hand side (Why?)

  **Hint**: left-hand side can reference to parts of objects on the heap – this is bad for most garbage collection algorithms

# Administrating Root Sets (1)

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

Call frame so far, without separated references:

```
METHOD m(a, b : Integer) : Integer IS
    c, d : Integer;
BEGIN
    RETURN 42;
END METHOD
```

| Address | Caller frame |
|---|---|
| R3 - 4 | SELF |
| R3-3 | a |
| R3-2 | b |
| R3-1 | Return address |
| R3 | Address of call frame |
| R3+1 | c |
| R2+2 | d |
| R3+3 | Return value |

# Administrating Root Sets (2)

… and with separation of references to the heap:

R2-Stack

| Address | Caller frame |
|---|---|
| : | Return address |
| : | Address of call frame |
| : | Intermediate values (L-values, values) |

R4-Stack

| Address | Caller frame |
|---|---|
| R3 – 3 | SELF |
| R3-2 | a |
| R3-1 | b |
| R3 | c |
| R3+1 | d |
| R3+2 | Return value |

# Simple Garbage Collection (1)

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

*Approach:*

- *Copy Collector*, using stack space
- Every object copies itself, and all objects it points to
- Most of the implementation is done in generated LOOP methods

# Simple Garbage Collection (2)

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

*Implementation*:

- Every class can generate a new object of its type, and can clone all its attributes
- *Object* contains a method that can clone the object, using the methods defined in derived classes
- A type-less („_Null“) attribute points to the copy of the object
- Every Object is copied only once

# Simple *Copying Collector* (1)

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

```
FOR EACH b root setDO
    b := b.lookupNewAddr;
END FOR
```

For every class

```
METHOD lookupNewAddr IS BEGIN
    IF newAddr = NULL THEN
        newAddr := NEW ThisClass;
        FOR EACH Reference r of SELF DO
            IF r # NULL THEN
                newAddr.r := r.lookupNewAddr;
            END IF
        END FOR
    END IF
    RETURN newAddr;
END METHOD
```

# Simple *Copying Collector (2)*

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

- Assumptions
  - The heap pointer is set to the new heap at the beginning so that *NEW* reserves space on the new heap
  - NEW initializes the attribute newAddr in the new object with *NULL*
- **Hints**
  - Do not forget attributes of base classes
  - *Integer* and *Boolean* contain an attribute that is not a reference

# Changing the Semantics of ":="

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

- **Problem**: During garbage collection , addresses of attributes can lie on the *R2*-stack
- Solution: evaluate right-hand side before left-hand side

```
CLASS Main IS
    a : Main;
    METHOD b IS
        a := NEW Main;
    END METHOD
END CLASS
```

R2-Stapel

| Address | Caller frame |
|---|---|
| R2 – 2 | Return address |
| R2 – 1 | Address of caller frame |
| R2 | Address of Self.a |

R4-Stapel

| Address | Caller frame |
|---|---|
| R3 - 1 | SELF |

# Bonus Tasks (1)

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

*Garbage Collection without using stack space:*

- True *Copy Collector*
  - No stack space consumption
    (depending on the structure of the heap)
- *Mark and Compact Collector*
  - Only one heap

5%

# Bonus Tasks (2)

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Universität Bremen

*Dispense of Variable Initialization (Idea):*

- Space on the stack for local variables is reserved only when they are updated
- An initialization corresponds to a *push* onto the *R4*-stack
- The relative address of local variables is determined by the order of their uses, not by the order of their declarations
- It is forbidden to use a local variable before it has been initialized

# Bonus Tasks (3)

*Dispense of Variable Initialization (Implementation):*

- *VarDeclaration* contains a label indicating whether the variable has been initialized

- Setting *VarDeclaration.offset* is delayed

5%

```
CLASS Main IS
    METHOD main IS
        a, b : Integer;
    BEGIN
        b := 5;
        a := b + a; | Fehler
    END METHOD
END CLASS
```

# *Much Success …* **(du courage!)**

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Universität Bremen

*… and nice vacations*