## **Semantics for Heaps**

Marco Vassena Philipp Hausmann Ondrej Pelech

## Research questions

- semantics for a minimal language
  - mutable heap
  - (exceptions)
- prove meta-theories

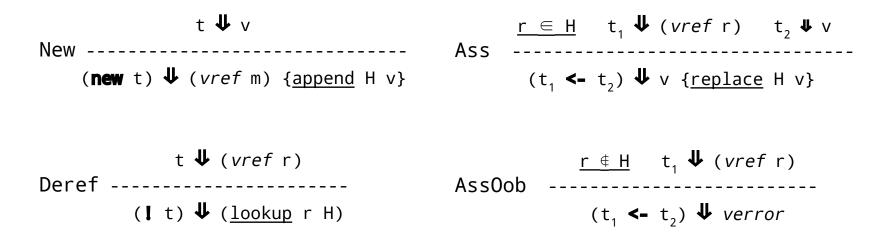
## Simple Typed Lambda Calculus

- Very verbose types
- Hard to guarantee totality
- Open research question

## Simple Heap Language

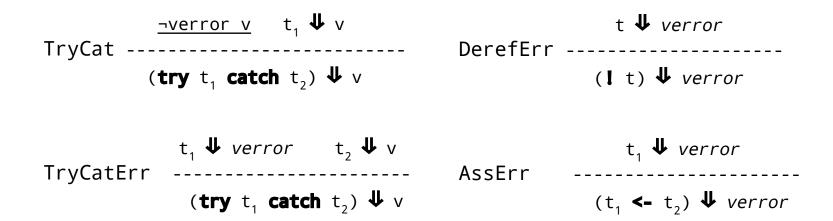
#### **Semantics**

Heap as a list of values, including error



#### **Semantics**

Handling errors



#### Research contribution

- Formal proofs
  - progress
  - preservation
  - o soundness
  - completeness
  - o determinism
  - type preservation in Heap

#### Research contribution

- Hoare logic
  - Hoare triples
  - Partial and total interpretation
  - reasoning about the correctness
  - proofs of the principal hoare logic theorems

#### **Further work**

- Extend hoare logic
  - statement vs expression
  - evaluation changes the heap
- Complete missing rules
- Make the theorems for hoare logic usable

#### References

- Types and Programming Languages, Benjamin C. Pierce
- Software Foundations and Programming Languages, Benjamin C. Pierce

# Thank you for your attention

Questions?

#### Research contribution

- Design and implementation in Agda
  - types
  - terms
  - values
  - heap

## The Types, Terms and Values

```
data Type : Set where
```

• • •

Boolean: Type

Ref : (ty : Type) -> Type

## The Types, Terms and Values

```
data Term : Type -> Set where
 . . .
 true : Term Boolean
 false
     : Term Boolean
     : ∀ {ty} -> Term ty
 error
 if_then_else_ : ∀ {ty} -> (cond : Term Boolean)
                 -> (tcase : Term ty)
                 -> (fcase : Term ty)
                 -> Term tv
          : ∀ {ty} -> Term ty -> Term (Ref ty)
 new
          : \forall {ty} -> Term (Ref ty) -> Term ty
 ref : ∀ {ty} -> N
                                        -> Term (Ref ty)
 try_catch_ : ∀ {ty} -> Term ty -> Term ty
```

### The Types, Terms and Values

## **Heap semantics**

```
data Heap : № -> Set where...
```

list of Values, including verror

## **Heap semantics**

```
E-New : BStep t \vee \rightarrow BStep {H2 = append H2 \vee} (new t) (vref m)
E-Deref : BStep t (vref r) \rightarrow BStep
                                                                (! t) (lookup r
H2)
E-Ass : (rep : Elem H3 r ty) \rightarrow
             BStep t1 (vref r) \rightarrow
             BStep t2 v
             BStep \{H2 = replace \ H3 \ rep \ v \} \ (t1 <- t2) \ v
E-TryCat : ¬ (isVError v)
            BStep t1
                                       V \rightarrow
             BStep (try t1 catch t2) v
```

## **Exception**

- raised when ref out of bounds
- error from Term
- verror from Value may be stored in Heap
- try\_catch\_ from Term