

# How much does it cost to normalize a proof ?

Simone Martini

Dipartimento di Scienze dell'Informazione  
*Alma mater studiorum* – Università di Bologna  
martini@cs.unibo.it

*Rencontre LIGC, Albano*  
14 décembre 26, 2007



# Outline

- 1 In limine: key concepts
- 2 The problem
- 3 Why this matters: ICC
- 4 Back to our problem
- 5 Challenges



# Outline

- 1 In limine: key concepts
- 2 The problem
- 3 Why this matters: ICC
- 4 Back to our problem
- 5 Challenges



## Disciplines: key concepts

Basic disciplines are structured around key, crucial (though non-exclusive) concepts:

- Physics: space, time
- Mathematics: number, symmetry, structure
- Chemistry: matter
- ...

They never exhaust them. . .

*Does Computer Science have such defining concepts?*



## Disciplines: key concepts

Basic disciplines are structured around key, crucial (though non-exclusive) concepts:

- Physics: space, time
- Mathematics: number, symmetry, structure
- Chemistry: matter
- ...

They never exhaust them...

Does *Computer Science* have such defining concepts?



## Of course it does!

- information
- effective (computation, procedure, process, ...)
- feasible
- abstraction hierarchy
- ...
  
- and the linguistic, *intensional* tools to support them



## Of course it does!

- information
- effective (computation, procedure, process, ...)
- feasible
- abstraction hierarchy
- ...
  
- and the linguistic, *intensional* tools to support them



## Of course it does!

- information
  - effective (computation, procedure, process, ...)
  - **feasible**
  - abstraction hierarchy
  - ...
- 
- **and the linguistic, *intensional* tools to support them**



# Outline

- 1 In limine: key concepts
- 2 The problem**
- 3 Why this matters: ICC
- 4 Back to our problem
- 5 Challenges



## The problem we will discuss

*How much does it cost to reduce  
a  $\lambda$ -term to normal form (if it exists) ?*

- Here *cost* is “*real cost*” on a standard machine,  
e.g. Turing machine
- Are there *natural* sensible measures for this cost?
- *I.e.* as a function of
  - ▶ the *type* of the term?
  - ▶ the *size* of the term?
  - ▶ the *length of the reduction* of the term?  
i.e., does polystep implies polytime?



# $\lambda$ -calculus

- Syntax:

$$M ::= x \mid (MM) \mid \lambda x.M$$

- Operational semantics:

$$(\lambda x.M)N \Rightarrow M[x \leftarrow N]$$

- This *is* the *copy rule* of procedure calls in Algol 60



# Outline

- 1 In limine: key concepts
- 2 The problem
- 3 Why this matters: ICC**
- 4 Back to our problem
- 5 Challenges



# Standard Computational Complexity

- Fix a **machine model** and a **cost** on it.
- Given a **program**  $P$  acting on data from set  $D$ , the *time complexity* of  $P$  is a function

$$T_P : \mathbb{N} \rightarrow \mathbb{N}$$

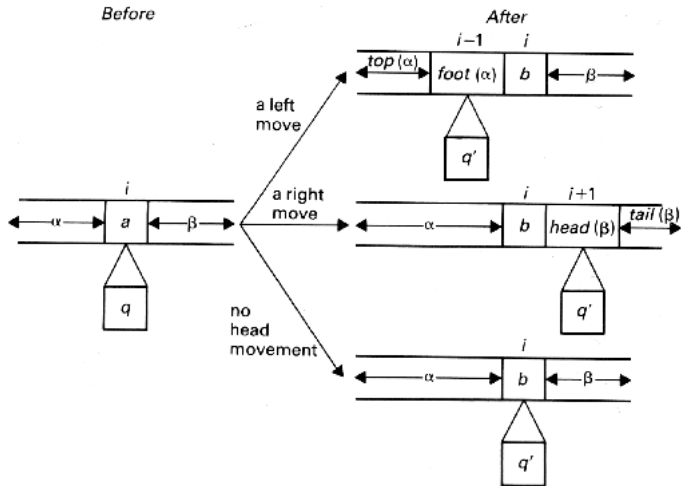
for each  $d \in D$ , the time needed to compute  $P(d)$  is less than  $T_P(|d|)$ .

- Given a **problem**  $Pb$  its *time complexity* is the time complexity of the most efficient program for solving  $Pb$ .
- **Reasonable** machine models are polytime related each other.



# Turing machine

A **finite** alphabet and a **finite** set of states



## ... and complexity classes

- $P_{\text{TIME}}$ : pbs solvable in polynomial time
- $FP_{\text{TIME}}$ : functions computable in polynomial time
- $EXPTIME$ : pbs solvable in exponential time
- $ELEMTIME$ : pbs solvable in Kalmar-elementary time
- $FELEMTIME$ : fncs computable in Kalmar-elementary time



## *Implicit Computational Complexity*

- A machine-free, logic-based investigation of the notion of **feasible computation**
- Investigate feasibility using **language restriction**, and not external measure conditions
- Import the logical “structure” into the linguistic level
- Many flavors.
- Here: Use **logical, proof-theoretical tools**



- **In the large:** use logic to study complexity classes
- **In the small:** use logic to study models of computation  
*i.e.*, machine-free models – no notion of constant-time step

*In some approaches to ICC,  
 $\lambda$ -calculus is at the core of both of them*



- **In the large:** use logic to study complexity classes
- **In the small:** use logic to study models of computation  
*i.e.*, machine-free models – no notion of constant-time step

*In some approaches to ICC,  
 $\lambda$ -calculus is at the core of both of them*



## Proof normalization...

$$\frac{\frac{[A] \quad \pi}{B} \quad \delta}{A \rightarrow B} \quad A}{B} \Rightarrow \frac{\delta}{A} \quad \pi \quad B$$



... is  $\beta$ -reduction

Gentzen-Prawitz notion of normalization is  $\beta$ -reduction in  $\lambda$ -calculus

$$\frac{\frac{[x : A]}{M : B} \quad \lambda x.M : A \rightarrow B}{(MN) : B} \quad N : A \quad \delta \quad \Rightarrow \quad M[x \leftarrow N] : B \quad \delta \quad \pi$$

$$(\lambda x.M)N \Rightarrow M[x \leftarrow N]$$



## ICC “in the large”

- After Girard's **light logics**...
- Many proof-theoretical characterizations of complexity classes

*Given a complexity class  $\mathcal{C}$ , find a logical system such that*

- **Soundness:**

*for any interesting type  $A$  and  $\pi : A \rightarrow A$ , there is a function  $f_A \in \mathcal{C}$  such that for any  $a \in A$ , the cost of normalizing  $\pi a$  is bounded by  $f_A(|a|)$ .*

- **Completeness:**

*for any  $F$  computable in complexity  $\mathcal{C}$ , there is a proof  $\pi_F$  “coding”  $F$ .*



## ICC “in the small”

Results on specific computational “machine-free” models.

*Do we have a natural implicit measure for the cost of reducing a  $\lambda$ -term, that is our  $\pi a$ , to normal form ?*

- The measure should be polynomially related to actual cost (“reasonable machine models”)
- In general, the answer is no



## ICC “in the small”

Results on specific computational “machine-free” models.

*Do we have a natural **implicit** measure for the cost of reducing a  $\lambda$ -term, that is our  $\pi a$ , to normal form ?*

- The measure should be **polynomially** related to actual cost (“reasonable machine models”)
- In general, the answer is **no**



# Outline

- 1 In limine: key concepts
- 2 The problem
- 3 Why this matters: ICC
- 4 Back to our problem**
- 5 Challenges



## Back to our problem

*Do we have a natural **implicit** measure for the cost of reducing a  $\lambda$ -term to normal form ?*

- The usual way:
  - ▶ bound the **length** of the reduction, **and**
  - ▶ bound the size of the **intermediate** terms, then
  - ▶ multiply them
- Explicit: mimicking a Turing machine analysis



## Count the steps

*Do we have a natural **implicit** measure for the cost of reducing a  $\lambda$ -term to normal form?*

- Naïve: just count the beta-steps
- The old problem: **duplication**
- When reducing

$$(\lambda x.M)N \Rightarrow M[x \leftarrow N]$$

in  $M$  there could be lots of  $x$ 's

- In general, it does **not** work
- However. . .



## Weak strategies

- Never normalize **under an abstraction**:  
 $\lambda x.M$  is always a normal form, no matter what is inside  $M$
- Important strategy:  
used by all functional programming languages
- Rational:  
interested in normal forms of “atomic” types  
(e.g., atomic values)



## Weak call by value

- Values:

$$V ::= x \mid \lambda x.M$$

- Weak call-by-value reduction:

$$\frac{}{(\lambda x.M)V \rightarrow_v M\{V/x\}} \qquad \frac{M \rightarrow_v N}{ML \rightarrow_v NL} \qquad \frac{M \rightarrow_v N}{LM \rightarrow_v LN}$$

$M$  are terms;  $V$  are values.

- $Time_v(M)$  is *the* number of beta-steps to normal form  
It is well defined



## *Efficient* implementations

- During the reduction of  $M$ , they manipulate subterms of the *original*  $M$
- They use *shared* representations for common subterms
- They use constant-time operations
- The number of such operations is poly (actually: linear) in the number of beta-steps (i.e.,  $Time_v(M)$ )



## The result

### Theorem

*There is a polynomial  $p : \mathbb{N}^2 \rightarrow \mathbb{N}$  such that for every lambda term  $M$ , a (shared version of the) normal form of  $M$  can be computed in time at most  $p(|M|, \text{Time}_v(M))$ .*

The *logical*, abstract notion of computation may be used as an actual cost measure.



## The result

### Theorem

*There is a polynomial  $p : \mathbb{N}^2 \rightarrow \mathbb{N}$  such that for every lambda term  $M$ , a (shared version of the) normal form of  $M$  can be computed in time at most  $p(|M|, \text{Time}_v(M))$ .*

The *logical*, abstract notion of computation may be used as an actual cost measure.



## Printing the result

- Sometimes the result is exponentially long in the input (e.g,  $n^2$ )
- The previous result could give us a **linear** complexity for obtaining a **shared** (compact) representation
- What if we want to actually have an explicit printout of the result?
- Use a different **cost model**:  
a step  $M \rightarrow N$  is accounted for  $\max(1, |N| - |M|)$



## Reasonable models

- Both models we described are *reasonable*
- We may *actually compute* on a Turing machine the normal form of  $M$  in that bound (modulo a poly)
- Any TM machine may be simulated by a lambda-term normalizing in the same time of the TM (modulo a poly)



# Outline

- 1 In limine: key concepts
- 2 The problem
- 3 Why this matters: ICC
- 4 Back to our problem
- 5 Challenges**



# Challenges, 1

- Implicit computational complexity is very fragmented; too many animals in the zoo. . .
- It is difficult to compare relative *intensional expressive power*. No general technique or result.

*Deep, foundational results are extremely needed.*



## Challenges, 2

- Absolute time
- Centralized model of computation
- No observer in the picture
- Where is this today's computer science?
  - ▶ Streaming of computations
  - ▶ Concurrent events
  - ▶ Environment interaction
  - ▶ Geographical decentralization
  - ▶ Scalability
  - ▶ ...



## Challenges, 2

- Absolute time
- Centralized model of computation
- No observer in the picture
- Where is this today's computer science?
  - ▶ Streaming of computations
  - ▶ Concurrent events
  - ▶ Environment interaction
  - ▶ Geographical decentralization
  - ▶ Scalability
  - ▶ ...



## The key concepts

- information
  - effective (computation, procedure, process, ...)
  - feasible
  - abstraction hierarchy
  - interaction
  - ...
- 
- and the linguistic, *intensional* tools to support them

