

# Proofs as *efficient* programs

Simone Martini

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

*Deduction, Computation, Experiment*  
April 3–4, 2007



# Outline

- 1 Proofs as programs
- 2 Implicit Computational Complexity, ICC
  - Light Logics
- 3 ICC: Tools and limitations



## Proofs of a theorem are not all alike

- Elegant vs awkward
- Short vs long
- Constructive vs non-constructive
  - ▶ By cardinality, there exist transcendental numbers
  - ▶ Liouville's proof that  $\sum_{k=1}^{\infty} 10^{-k!}$  is transcendental



## The formal perspective: *normal* proofs

- Compare and study *formal* proofs
- In *normal* form
- Narrow approach: computer science perspective



## The formal perspective: *normal* proofs

- Compare and study *formal* proofs
- In *normal* form
- Narrow approach: computer science perspective



## Proofs and programs

- After Gentzen, Curry, Prawitz, Howard, Lambeck, ...
- Curry: Hilbert-style systems
  - ▶ Intuitionistically valid propositions correspond to types of certain lambda-terms
  - ▶ The deduction theorem corresponds to Curry's abstraction algorithm
    - ★  $A \vdash B$  iff  $\vdash A \rightarrow B$
    - ★ Given an expression  $b$  of type  $B$  with free var  $x$  of type  $A$ , construct an expression  $[x]b$  of type  $A \rightarrow B$  behaving as follows

$$([x]b)a \Rightarrow b[x \leftarrow a]$$



## Proofs and programs

- After Gentzen, Curry, Prawitz, Howard, Lambeck, ...
- Curry: Hilbert-style systems
  - ▶ Intuitionistically valid propositions correspond to types of certain lambda-terms
  - ▶ The deduction theorem corresponds to Curry's abstraction algorithm
    - ★  $A \vdash B$  iff  $\vdash A \rightarrow B$
    - ★ Given an expression  $b$  of type  $B$  with free var  $x$  of type  $A$ , construct an expression  $[x]b$  of type  $A \rightarrow B$  behaving as follows

$$([x]b)a \Rightarrow b[x \leftarrow a]$$



# Proofs as programs

LOGIC	COMPUTATION
Formula	Type
Proof	Program
Normal proof	Data



# The Curry-Howard correspondence: Annotated proofs

Two different worlds ( $\lambda$ -calculus and intuitionistic logic) become the same...

$$x : A \vdash x : A$$

$$\frac{\Gamma \vdash M : C}{\Gamma, x : A \vdash M : C} \text{ (Weak.)}$$

$$\frac{\Gamma, y : A, z : A \vdash M : B}{\Gamma, x : A \vdash M[z/x, y/z] : B} \text{ (Contr.)}$$

$$\frac{\Gamma \vdash M : A \rightarrow \Delta \vdash N : B}{\Gamma \Delta \vdash MN : B} \text{ (}\rightarrow, E\text{)}$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x. M : A \rightarrow B} \text{ (}\rightarrow, I\text{)}$$



# Normalization

Gentzen-Prawitz notion of normalization is  $\beta$ -reduction in  $\lambda$ -calculus, or is the *copy rule* of procedure calls in Algol 60

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



# The Curry-Howard correspondence: Computing with proofs

- **Notation** for proofs:  $\lambda$ -terms.
- **Normalization** of proofs is  $\beta$ -reduction on  $\lambda$ -terms
- Standard proof-theoretical question:  
How much does it cost to normalize a proof?
- *Technically*: How big is the normal form of a proof?  
(As a function of the size of the original proof.)

## Theorem

*In any reasonable logic,  
if  $\pi$  normalizes to  $\pi'$ ,  $|\pi'|$  is (hyper-)exponential in  $|\pi|$*



# The Curry-Howard correspondence: Computing with proofs

- **Notation** for proofs:  $\lambda$ -terms.
- **Normalization** of proofs is  $\beta$ -reduction on  $\lambda$ -terms
- Standard proof-theoretical question:  
How much does it cost to normalize a proof?
- *Technically*: How big is the normal form of a proof?  
(As a function of the size of the original proof.)

## Theorem

*In any reasonable logic,  
if  $\pi$  normalizes to  $\pi'$ ,  $|\pi'|$  is (hyper-)exponential in  $|\pi|$*



# The Curry-Howard correspondence: Computing with proofs

- **Notation** for proofs:  $\lambda$ -terms.
- **Normalization** of proofs is  $\beta$ -reduction on  $\lambda$ -terms
- Standard proof-theoretical question:  
How much does it cost to normalize a proof?
- *Technically*: How big is the normal form of a proof?  
(As a function of the size of the original proof.)

## Theorem

*In any reasonable logic,  
if  $\pi$  normalizes to  $\pi'$ ,  $|\pi'|$  is (hyper-)exponential in  $|\pi|$*



## A change of perspective

- (If the logic is powerful enough) we can give formulas representing *significant datatypes*:

*Integers, Booleans, Binary strings, Lists, Trees, ...*

- Proofs of the form

$$\pi : \ulcorner \text{Integers} \urcorner \rightarrow \ulcorner \text{Integers} \urcorner$$

or

$$\pi' : \ulcorner \text{Binary string} \urcorner \rightarrow \ulcorner \text{Binary string} \urcorner$$

are programs manipulating integers, or strings

- If  $w$  is a  $\ulcorner \text{Binary string} \urcorner$

$$\pi' w$$

gives a computation normalizing on a  $\ulcorner \text{Binary string} \urcorner$



## Change of perspective, 2

- Given a proof (program)

$$\pi : A \rightarrow B$$

study how much does it cost to normalize

$$\pi w$$

as a function of  $|w|$  (i.e., when  $w$  varies in  $A$ ).



# Complexity

- Given a **program**  $P$  acting on data from set (type)  $D$ , the *time complexity* of  $P$  is a function

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

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

- Given a **problem**  $Pb$  its *time complexity* is the time complexity of the most efficient program for solving  $Pb$ .



## ... and complexity classes

- **P**TIME: pbs solvable in polynomial time
- **F**P**T**IME: functions computable in poly time
- **E**X**P**TIME: pbs solvable in exponential time
- **E**L**E**M**T**IME: pbs solvable in Kalmar elementary time
- **F**E**L**E**M**TIME: functions computable in Kalmar elementary time



# Implicit Computational Complexity

- Giving machine-free, logic-based characterizations of complexity classes
  - ▶ Model Theory (Finite Model Theory);
  - ▶ Recursion Theory;
  - ▶ **Proof Theory** (via Curry-Howard):

*Given a complexity class  $\mathcal{C}$ , find a logical system  $LS$ , 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"  $f$ .*



# Implicit Computational Complexity

- Giving machine-free, logic-based characterizations of complexity classes
  - ▶ Model Theory (Finite Model Theory);
  - ▶ Recursion Theory;
  - ▶ **Proof Theory** (via Curry-Howard):

*Given a complexity class  $\mathcal{C}$ , find a logical system  $LS$ , 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”  $f$ .*



## Logic with limited complexity normalization

- Exponential blow-up caused by arbitrary duplication (i.e., contraction)
- Look for substructural logics  
*too limited expressivity*
- Reintroduce controlled duplication: Linear logic  
*too complex normalization*
- Drastically limit duplication, still allowing for some  
*Light logics* (Girard, 1998)



## Logic with limited complexity normalization

- Exponential blow-up caused by arbitrary duplication (i.e., contraction)
- Look for substructural logics  
*too limited expressivity*
- Reintroduce controlled duplication: Linear logic  
*too complex normalization*
- Drastically limit duplication, still allowing for some  
*Light logics* (Girard, 1998)



## Logic with limited complexity normalization

- Exponential blow-up caused by arbitrary duplication (i.e., contraction)
- Look for substructural logics  
*too limited expressivity*
- Reintroduce controlled duplication: Linear logic  
*too complex normalization*
- Drastically limit duplication, still allowing for some  
*Light logics* (Girard, 1998)



## Logic with limited complexity normalization

- Exponential blow-up caused by arbitrary duplication (i.e., contraction)
- Look for substructural logics  
*too limited expressivity*
- Reintroduce controlled duplication: Linear logic  
*too complex normalization*
- Drastically limit duplication, still allowing for some  
*Light logics* (Girard, 1998)



## Logic with limited complexity normalization

- Exponential blow-up caused by arbitrary duplication (i.e., contraction)
- Look for substructural logics  
*too limited expressivity*
- Reintroduce controlled duplication: Linear logic  
*too complex normalization*
- Drastically limit duplication, still allowing for some  
*Light logics* (Girard, 1998)



## Logic with limited complexity normalization

- Exponential blow-up caused by arbitrary duplication (i.e., contraction)
- Look for substructural logics  
*too limited expressivity*
- Reintroduce controlled duplication: Linear logic  
*too complex normalization*
- Drastically limit duplication, still allowing for some  
*Light logics* (Girard, 1998)



# I(ME)LL

$$A \vdash A \text{ (Ax)}$$

$$\frac{\Gamma \vdash A \quad A, \Delta \vdash B}{\Gamma, \Delta \vdash B} \text{ (Cut)}$$

$$\frac{\Gamma \vdash C}{\Gamma, !A \vdash C} \text{ (Weak.)}$$

$$\frac{\Gamma, !A, !A \vdash B}{\Gamma, !A \vdash B} \text{ (Contr.)}$$

$$\frac{\Gamma \vdash A \quad B, \Delta \vdash C}{\Gamma, A \multimap B, \Delta \vdash C} \text{ (}\multimap, l\text{)}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B} \text{ (}\multimap, r\text{)}$$

$$\frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C} \text{ (}\otimes, l\text{)}$$

$$\frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B} \text{ (}\otimes, r\text{)}$$

$$\frac{A_1, \dots, A_n \vdash B}{!A_1, \dots, !A_n \vdash !B} \text{ (!)}$$

$$\frac{\Gamma, A \vdash B}{\Gamma, !A \vdash B} \text{ (}\epsilon\text{)}$$

$$\frac{\Gamma, !!A \vdash B}{\Gamma, !A \vdash B} \text{ (}\delta\text{)}$$

$$\frac{\Gamma, T[S/t] \vdash C}{\Gamma, \forall t. T \vdash C} \text{ (}\forall, l\text{)}$$

$$\frac{\Gamma \vdash C}{\Gamma \vdash \forall t. C} \text{ } t \notin FV(\Gamma) \text{ (}\forall, r\text{)}$$



## Fine control of duplication

- How are we allowed to use the duplicated resources (i.e., !-marked formulas)?
- Write  $A \equiv B$  for  $A \multimap B$  and  $B \multimap A$
- The most fundamental property is  $!A \equiv !A \otimes !A$
- It is obtained from rules (C), (W) and (!)
- But we have more properties...
  - $!A \multimap A$ , from ( $\epsilon$ ) (“dereliction”)
  - $!A \multimap !!A$ , from ( $\delta$ ) (“digging”)
  - The interplay between these rules is the main source for complexity of normalization and expressivity
  - From a modal logic perspective: ! in LL is like  $\Box$  in modal logic S4. . .



## Fine control of duplication

- How are we allowed to use the duplicated resources (i.e., !-marked formulas)?
- Write  $A \equiv B$  for  $A \multimap B$  and  $B \multimap A$
- The most fundamental property is  $!A \equiv !A \otimes !A$
- It is obtained from rules (C), (W) and (!)
- But we have more properties...
- $!A \multimap A$ , from ( $\epsilon$ ) (“dereliction”)
- $!A \multimap !!A$ , from ( $\delta$ ) (“digging”)
- The interplay between these rules is the main source for complexity of normalization and expressivity
- From a modal logic perspective: ! in LL is like  $\Box$  in modal logic S4. . .



## Subsystems of Linear Logic

	$!A \multimap !!A$	$!A \multimap A$	$!A \cong !A \otimes !A$
<b>ELL</b>	NO	NO	YES
<b>LLL</b>	NO	NO	YES
<b>SLL</b>	NO	$!A \multimap A \otimes \dots \otimes A$	

...and their expressive power

<b>ELL</b>	Elementary Time
<b>LLL</b>	Polynomial Time
<b>SLL</b>	Polynomial Time



## Subsystems of Linear Logic

	$!A \multimap !!A$	$!A \multimap A$	$!A \cong !A \otimes !A$
<b>ELL</b>	NO	NO	YES
<b>LLL</b>	NO	NO	YES
<b>SLL</b>	NO	$!A \multimap A \otimes \dots \otimes A$	

...and their expressive power

<b>ELL</b>	Elementary Time
<b>LLL</b>	Polynomial Time
<b>SLL</b>	Polynomial Time



## Subsystems of Linear Logic, II

As rules:

	(!)	( $\delta$ )	( $\epsilon$ )	(C)	(mplex)	(u!)
<b>ELL</b>	YES	NO	NO	YES	NO	derivable
<b>LLL</b>	NO	NO	NO	YES	NO	YES + (§)
<b>SLL</b>	YES	NO	NO	NO	YES	derivable

where

$$\frac{\Gamma, !A, !A \vdash B}{\Gamma, !A \vdash B} \text{ (C)} \qquad \frac{A_1, \dots, A_n \vdash B}{!A_1, \dots, !A_n \vdash !B} \text{ (!)}$$

$$\frac{\Gamma, A \vdash B}{\Gamma, !A \vdash B} \text{ ( $\epsilon$ )} \qquad \frac{\Gamma, !!A \vdash B}{\Gamma, !A \vdash B} \text{ ( $\delta$ )}$$

$$\frac{\Gamma, A, A \vdash C}{\Gamma, !A \vdash C} \text{ (mplex)} \qquad \frac{A \vdash C}{!A \vdash !C} \text{ u!}$$



## Shape of the results

Take Light Linear Logic, LLL.

### Theorem

Let  $\pi : \ulcorner \text{Binary string} \urcorner \rightarrow \ulcorner \text{Binary string} \urcorner$ .

For any  $w : \ulcorner \text{Binary string} \urcorner$ ,

$$\pi w$$

normalizes in time proportional to  $(d + 1)|\pi w|^{2d}$ , where  $d$  depends only on  $\pi$ .



## The zoo of “light” logics

- Polytime: LLL, LAL, SLL, SAL, (BLL)
- Elementary time: ELL, EAL
- Log-space: (Stratified BLL)

Always *ad hoc* proofs of soundness and completeness.

No “structural”, uniform ways for translating other formalisms for polytime.

Some of these logics have an interest for their own, but:

*Can we make order in this situation?*



## The zoo of “light” logics

- Polytime: LLL, LAL, SLL, SAL, (BLL)
- Elementary time: ELL, EAL
- Log-space: (Stratified BLL)

Always *ad hoc* proofs of soundness and completeness.

No “structural”, uniform ways for translating other formalisms for polytime.

Some of these logics have an interest for their own, but:

*Can we make order in this situation?*



# Algebraic context semantics

Dal Lago, 2005 and ff

- General technique for proving quantitative results
- Simple variant of “*Geometry of interaction*”
- Attach weights to graphs representing terms
- Weights depend on the size of the reducts of a term

*The technique works for a variety of systems:*

- *Gödel's T (2005),*
- *ELL, SLL, LLL and their affine variants (2006),*
- *“Optimal reduction” (2007)*



# Algorithms

*Ma fin est mon commencement*

- Proofs of a theorem are not all alike
- We have systems for polytime proofs
- Do they admit *simple, intuitive* proofs (*programs*)?
- Extensional completeness for a complexity class  
vs  
*Intensional* expressivity

*Which algorithms can be expressed in the system?*



# Algorithms

*Ma fin est mon commencement*

- Proofs of a theorem are not all alike
- We have systems for polytime proofs
- Do they admit simple, intuitive proofs (*programs*)?
- Extensional completeness for a complexity class  
vs  
*Intensional* expressivity

*Which algorithms can be expressed in the system?*



# Algorithms

*Ma fin est mon commencement*

- Proofs of a theorem are not all alike
- We have systems for polytime proofs
- Do they admit **simple, intuitive** proofs (*programs*)?
- Extensional completeness for a complexity class  
vs  
*Intensional* expressivity

*Which algorithms can be expressed in the system?*



## Limited expressivity

- Natural polytime algorithms are *not* expressible in light logics
- They lack the right (simple) inductive structure
- Light logics do not allow *benign sharing*
- How can we extend those formalisms?



## Challenges

- 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.
- It is not usually the case that a system can be *extended* with new features preserving its quantitative properties

*Deep, foundational results are extremely needed.*

