

A Language for Mathematics

Mohan Ganesalingam

University of Cambridge, Computer Laboratory
mg262 at cam.ac.uk

March 10, 2009

Long-Term Project

- ▶ Joint work with Thomas Barnet-Lamb.
- ▶ Aim to have computers do maths in the way mathematicians do.
- ▶ First step: read and write maths like mathematicians.
- ▶ Only discuss linguistic/computational issues here.

- ▶ Overall Aim: Build a computer language for real mathematics.
 - ▶ As close to actual maths as possible.
 - ▶ But still an artificial language, not an attempt to parse unrestricted text.
 - ▶ Deterministic.
- ▶ Part I: Describe the language of mathematics and situate it among formal and natural languages.
- ▶ Part II: Outline the architecture of a computer language for mathematics.

Part I

The Language of Mathematics

Overview of Part I

- ▶ Overall Aim: Build a computer language for real mathematics.
 - ▶ Part I Aim: Describe the language of mathematics and situate it among formal and natural languages.
-

1. Examples of Real Mathematics
2. Similarities between Mathematics and Formal Languages
3. Mathematics is not a Conventional Computer Language
4. Mathematics as a Natural Language
5. Mathematics as a Linguistic Domain
6. A “Natural Formal Language”

There are infinitely many prime numbers.

Every even integer greater than 2 is the sum of two prime numbers.

Let c be a simple closed curve in the plane \mathbb{R}^2 . Then the complement of the image of c consists of two distinct connected components. One of these components is bounded and the other is unbounded.

Sylow's Theorems

Let G be a finite group whose order is divisible by the prime p . Suppose p^m is the highest power of p which is a factor of $|G|$ and set

$$k = \frac{|G|}{p^m}.$$

Then

1. the group G contains at least one subgroup of order p^m ,
2. any two subgroups of G of order p^m are conjugate, and
3. the number of subgroups of G of order p^m is congruent to 1 modulo p and is a factor of k .

Similarities between Mathematics and Formal Languages

- ▶ Variables

“Let G be a finite group ...”

- ▶ Type

“ $A + B$ ”

Vector, Set, Matrix, Group, ...

- ▶ Precise Semantics

Unambiguously translatable into a logic.

Standardised Notion of a Formal Computer Language

1. Unambiguous Syntax. (Typically, LR(1).)

▶ Parsed in linear time.

2. Precedence.

$$a + b * c$$

3. Standardised Type Systems.

4. Semantics in Mainstream Logic.

Standardised Notion of a Formal Computer Language

1. Unambiguous Syntax. (Typically, LR(1).)

▶ Parsed in linear time.

2. Precedence.

$$a + (b * c)$$

3. Standardised Type Systems.

4. Semantics in Mainstream Logic.

theorem :: *GROUP_10:12*

for G **being** finite Group,
p **being** prime (natural number)
holds ex P **being** Subgroup of G **st**
P is_Sylow_p-subgroup_of_prime p;

theorem :: *GROUP_10:14*

for G **being** finite Group,
p **being** prime (natural number) **holds**
(**for** H **being** Subgroup of G **st**
H is_p-group_of_prime p **holds**
ex P **being** Subgroup of G **st**
P is_Sylow_p-subgroup_of_prime p
...

[MIZAR: (Trybulec, 1972). Excerpt: (Wiedijk, 2008).]

Sylow's Theorems

Let G be a finite group whose order is divisible by the prime p . Suppose p^m is the highest power of p which is a factor of $|G|$ and set

$$k = \frac{|G|}{p^m}.$$

Then

1. the group G contains at least one subgroup of order p^m ,
2. any two subgroups of G of order p^m are conjugate, and
3. the number of subgroups of G of order p^m is congruent to 1 modulo p and is a factor of k .

```
theorem :: GROUP_10:12
  for G being finite Group,
  p being prime (natural number)
  holds ex P being Subgroup of G st
    P is_Sylow_p-subgroup_of_prime p;
```

```
theorem :: GROUP_10:14
  for G being finite Group,
  p being prime (natural number) holds
  (for H being Subgroup of G st
    H is_p-group_of_prime p holds
    ex P being Subgroup of G st
      P is_Sylow_p-subgroup_of_prime p
      & H is Subgroup of P) &
  (for P1,P2 being Subgroup of G
    st P1 is_Sylow_p-subgroup_of_prime p
    & P2 is_Sylow_p-subgroup_of_prime p
    holds P1,P2 are_conjugated);
```

```
theorem :: GROUP_10:15
  for G being finite Group,
  p being prime (natural number) holds
  card the_sylow_p-subgroups_of_prime(p,G)
  mod p = 1 &
  card the_sylow_p-subgroups_of_prime(p,G)
  divides ord G;
```

1. Full Context-Free Syntax

Mathematics isn't a Conventional Formal Language

1. Full Context-Free Syntax
2. **'Donkey Sentence'**

Compositionality

Compositionality: Meaning of the whole is a function of the meanings of the parts.

$\text{trans}(\text{'Alice is happy'}) = \text{happy}(\text{Alice})$

$\text{trans}(\text{'Bob is sad'}) = \text{sad}(\text{Bob})$

$\text{trans}(S_1 \text{ and } S_2) = \text{trans}(S_1) \wedge \text{trans}(S_2)$

$\text{trans}(\text{'Alice is happy and Bob is sad'}) = \text{happy}(\text{Alice}) \wedge \text{sad}(\text{Bob})$

'Donkey Sentence'

- ▶ Then $V = U \cap H$ for some U in \mathcal{T} , by definition of \mathcal{T}_H , and $U \cap H = i^{-1}(U)$, so $g^{-1}(V) = g^{-1}(i^{-1}(U)) = (i \circ g)^{-1}(U)$.

Sutherland, W. A., *Introduction to Metric and Topological Spaces*, OUP 1975, p. 52.

'Donkey Sentence'

- ▶ Then $V = U \cap H$ for some U in \mathcal{T} , by definition of \mathcal{T}_H , and $U \cap H = i^{-1}(U)$, so $g^{-1}(V) = g^{-1}(i^{-1}(U)) = (i \circ g)^{-1}(U)$.

Sutherland, W. A., *Introduction to Metric and Topological Spaces*, OUP 1975, p. 52.

'Donkey Sentence'

- ▶ Then $V = U \cap H$ for some U in \mathcal{T} and $U \cap H = i^{-1}(U)$.

'Donkey Sentence'

- Then $\underbrace{V = U \cap H}_{\alpha[U]}$ for some U in \mathcal{T} and $\underbrace{U \cap H = i^{-1}(U)}_{\beta[U]}$.

'Donkey Sentence'

- ▶ Then $\underbrace{V = U \cap H}_{\alpha[U]}$ for some U in \mathcal{T} and $\underbrace{U \cap H = i^{-1}(U)}_{\beta[U]}$.
- ▶ $\alpha[U]$ for some U in \mathcal{T} and $\beta[U]$.

'Donkey Sentence'

- ▶ Then $\underbrace{V = U \cap H}_{\alpha[U]}$ for some U in \mathcal{T} and $\underbrace{U \cap H = i^{-1}(U)}_{\beta[U]}$.
- ▶ $(\alpha[U]$ for some U in $\mathcal{T})$ and $\beta[U]$.

'Donkey Sentence'

- ▶ Then $\underbrace{V = U \cap H}_{\alpha[U]}$ for some U in \mathcal{T} and $\underbrace{U \cap H = i^{-1}(U)}_{\beta[U]}$.
- ▶ $(\alpha[U]$ for some U in $\mathcal{T})$ and $\beta[U]$.

'Donkey Sentence'

- ▶ Then $\underbrace{V = U \cap H}_{\alpha[U]}$ for some U in \mathcal{T} and $\underbrace{U \cap H = i^{-1}(U)}_{\beta[U]}$.
- ▶ $(\alpha[U]$ for some U in $\mathcal{T})$ and $\beta[U]$.
- ▶ Compositional Analysis in First-order Logic:

'Donkey Sentence'

- ▶ Then $\underbrace{V = U \cap H}_{\alpha[U]}$ for some U in \mathcal{T} and $\underbrace{U \cap H = i^{-1}(U)}_{\beta[U]}$.
- ▶ $(\alpha[U] \text{ for some } U \text{ in } \mathcal{T})$ and $\beta[U]$.
- ▶ Compositional Analysis in First-order Logic:
- ▶ $\exists U. (in(U, \mathcal{T}) \wedge \hat{\alpha}[U])$ $\hat{\beta}[U]$

'Donkey Sentence'

- ▶ Then $\underbrace{V = U \cap H}_{\alpha[U]}$ for some U in \mathcal{T} and $\underbrace{U \cap H = i^{-1}(U)}_{\beta[U]}$.
- ▶ $(\alpha[U] \text{ for some } U \text{ in } \mathcal{T})$ and $\beta[U]$.
- ▶ Compositional Analysis in First-order Logic:
- ▶ $\exists U. (in(U, \mathcal{T}) \wedge \hat{\alpha}[U]) \quad \wedge \quad \hat{\beta}[U]$

'Donkey Sentence'

▶ Then $\underbrace{V = U \cap H}_{\alpha[U]}$ for some U in \mathcal{T} and $\underbrace{U \cap H = i^{-1}(U)}_{\beta[U]}$.

▶ $(\alpha[U]$ for some U in $\mathcal{T})$ and $\beta[U]$.

▶ Compositional Analysis in First-order Logic:

▶ $\exists U.(in(U, \mathcal{T}) \wedge \hat{\alpha}[U]) \wedge \hat{\beta}[U]$
↑
unbound variable

Mathematics isn't a Conventional Formal Language

1. Full Context-Free Syntax
2. 'Donkey Sentence'
3. **Adaptivity**

- ▶ The syntax of mathematics expands as more mathematics is encountered.
- ▶ In particular, *definitions* introduce new syntax (and corresponding semantics).

Definition. If G is a group and $a \in G$, then the **centralizer** of a in G is the set of all $a \in G$ that commute with a ; it is denoted by $C_G(a)$.

- ▶ This definition adds the notation ' $C_{\bullet}(\bullet)$ ' to the grammar of the language.

Mathematics isn't a Conventional Formal Language

1. Full Context-Free Syntax
2. 'Donkey Sentence'
3. Adaptivity
4. **Inadequacy of Precedence**

Inadequacy of Precedence

- ▶ Example with Symbols

Arithmetic:

$$A + B = C$$

Lambda Calculus:

$$\lambda + \mathbf{M} = \mathbf{N}$$

Inadequacy of Precedence

- ▶ Example with Symbols

Arithmetic:

$$(A + B) = C$$

Lambda Calculus:

$$\lambda + (\mathbf{M} = \mathbf{N})$$

Inadequacy of Precedence

- ▶ Example with Symbols

Arithmetic:

$$(A + B) = C$$

Lambda Calculus:

$$\lambda + (\mathbf{M} = \mathbf{N})$$

- ▶ Example with Words

Two Sentences of the form 'P **and** Q **for** X':

So $\underbrace{f(x) > 0 \text{ for } x > 0}_P$ **and** $\underbrace{f(x) < 0 \text{ for } x < 0}_Q$ **for** $\underbrace{x < 0}_X$.

Whenever \underbrace{aRb}_P **and** \underbrace{bRc}_Q **for** $\underbrace{\text{some } a, b, c}_X$ then aRc .

Inadequacy of Precedence

- ▶ Example with Symbols

Arithmetic:

$$(A + B) = C$$

Lambda Calculus:

$$\lambda + (\mathbf{M} = \mathbf{N})$$

- ▶ Example with Words

Two Sentences of the form 'P **and** Q **for** X':

So $\underbrace{f(x) > 0 \text{ for } x > 0}_P$ **and** $\underbrace{(f(x) < 0 \text{ for } x < 0)}_Q$ **for** $\underbrace{x < 0}_X$.

Whenever $\underbrace{(aRb)}_P$ **and** $\underbrace{(bRc)}_Q$ **for** $\underbrace{\text{some } a, b, c}_X$ then aRc .

Mathematics isn't a Conventional Formal Language

1. Full Context-Free Syntax
2. 'Donkey Sentence'
3. Adaptivity
4. Inadequacy of Precedence
5. **Type-Dependent Syntax**

Type-Dependent Syntax

- ▶ Consider:

Suppose that $K + f = x \in \mathcal{S}$, where K is an integer and $f \in [0, 1)$.

- ▶ ' $K + f = x$ ' is a priori syntactically ambiguous.
 - ▶ ' $(K + f) = x$ '
 - ▶ or ' $K + (f = x)$ '?
- ▶ The types of x , K , F and \mathcal{S} are needed to parse the sentence correctly.
- ▶ (And we need to parse the sentence to determine the types of K and F ! Discussed further in Part II.)

Mathematics isn't a Conventional Formal Language

1. Full Context-Free Syntax
2. 'Donkey Sentence'
3. Adaptivity
4. Inadequacy of Precedence
5. Type-Dependent Syntax

1. Presupposition

Presupposition

- ▶ “the quotient of a and b”
 - ▶ Presupposes that this quotient exists.



$$k = \frac{|G|}{p^m}$$

- ▶ Symbolic equivalent of the ‘quotient’ example.
- ▶ Presupposes that p^m is nonzero.

1. Presupposition
2. **Anaphora**

Let c be a simple closed curve in the plane \mathbb{R}^2 . Then the complement of the image of c consists of two distinct connected components. **One of these components** is bounded and **the other** is unbounded.

1. Presupposition
2. Anaphora
3. **Generalised Quantification**

- ▶ **'Any two** subgroups of G of order p^m are conjugate.'
- ▶ Compare MIZAR equivalent:
(for P1,P2 being Subgroup of G
st P1 is_Sylow_p-subgroup_of_prime p
& P2 is_Sylow_p-subgroup_of_prime p
holds P1,P2 are_conjugated);
- ▶ **'The** group G contains **at least one** subgroup of order p^m .'

Mathematics as a Natural Language

1. Presupposition
2. Anaphora
3. Generalised Quantification
4. **Ellipsis**

- ▶ 'Let V be a vector space.'
- = 'Let V be a vector space over some field k .'
- ▶ 'the domain'
- = 'the domain of the function f '
- ▶ ' gh '
- = $*_G(g, h)$ where the group G is not explicitly specified.

1. Presupposition
2. Anaphora
3. Generalised Quantification
4. Ellipsis
5. **Non-Local Disambiguation**

Non-Local Disambiguation

Suppose that $K + f = x \in \mathcal{S}$, where K is an integer and $f \in [0, 1)$.

- ▶ In order to parse ' $K + f = x \in \mathcal{S}$ ', we need information found further on in the sentence.
- ▶ We can need information arbitrarily far away in the sentence:

Suppose that $K + f = x \in \mathcal{S}$, **and P, and also Q, and ...**, where K is an integer and $f \in [0, 1)$.

- ▶ Cf. garden path sentences in natural languages.
 - ▶ 'The horse raced past the barn fell.'

Mathematics as a Natural Language

1. Presupposition
2. Anaphora
3. Generalised Quantification
4. Ellipsis
5. Non-Local Disambiguation

There are infinitely many prime numbers.

Every even integer greater than 2 is the sum of two prime numbers.

Let c be a simple closed curve in the plane \mathbb{R}^2 . Then the complement of the image of c consists of two distinct connected components. One of these components is bounded and the other is unbounded.

Mathematics as a Linguistic Domain

- ▶ Mathematical terms defined; few extra-mathematical terms.
- ▶ Limited inflectional morphology.
- ▶ Convoluted syntax is rare.
- ▶ Clean, total semantics. (Suitable input to a program.)
- ▶ Language starts with a small core, expands via definitions; definitions contain all syntactic and semantic information.
--> Benefits of both closed domains and open domains.

A “Natural Formal Language”

- ▶ Throw away preconceptions about computer languages.
- ▶ Combine linguistics and theories of formal languages...
- ▶ ... and add some new theory.
- ▶ Formal language and natural language: each overcomes difficulties arising from the other.

Overview of Part I

- ▶ Overall Aim: Build a computer language for real mathematics.
 - ▶ Part I Aim: Describe the language of mathematics and situate it among formal and natural languages.
-

1. Examples of Real Mathematics
2. Similarities between Mathematics and Formal Languages
3. Mathematics is not a Conventional Computer Language
4. Mathematics as a Natural Language
5. Mathematics as a Linguistic Domain
6. A “Natural Formal Language”

Part II

A Language for Mathematics

Overview of Part II

- ▶ Overall Aim: Build a computer language for real mathematics.
- ▶ Part II Aim: Outline the architecture of a computer language for mathematics.

-
1. Lexicon
 2. Syntax
 3. Semantics
 4. Type
 5. Adaptivity

- ▶ Cannot handle unrestricted lexicon.
- ▶ Mathematical terms are always defined:

Definition. A **group** is a **semigroup** $(G, *)$ containing an **element** e such that ...

- ▶ Extra-mathematical terms: adapt notion of a fragment.

| | | | | |
|--------|-----|------|---------|------------|
| a | all | also | always | an |
| and | any | are | as | as desired |
| assume | at | be | because | both |
| but | by | call | called | ... |

Sylow's Theorems.

Let G be a finite group whose order is divisible by the prime p . Suppose p^m is the highest power of p which is a factor of $|G|$ and set

$$k = \frac{|G|}{p^m}.$$

Then

1. the group G contains at least one subgroup of order p^m ,
2. any two subgroups of G of order p^m are conjugate, and
3. the number of subgroups of G of order p^m is congruent to 1 modulo p and is a factor of k .

Theorem (“Sylow’s Theorems”).

Let G be a finite group whose order is divisible by a prime p . Let m be the integer s.t. p^m is the largest power of p which divides $|G|$ and set

$$k = |G|/p^m.$$

Then

1. the group G contains a subgroup of order p^m ,
2. any two subgroups of G of order p^m are conjugate, and
3. the number of subgroups of G which have order p^m is congruent to 1 modulo p and is a factor of k .

Let G be a finite group whose order is divisible by a prime p . Let m be the integer s.t. p^m is the highest power of p which divides $|G|$ and set

$$k = |G|/p^m.$$

Then

1. the group G contains a subgroup of order p^m ,
2. any two subgroups of G of order p^m are conjugate, and
3. the number of subgroups of G which have order p^m is congruent to 1 modulo p and is a factor of k .

- ▶ Grammar consists of context-free rules.
- ▶ Lexical items encoded as rules.
- ▶ Example excerpt:

$$S \rightarrow NP_{sg} \quad VP_{sg}$$

0 1

$$NP_{sg} \rightarrow Det_{sg} \quad NBar$$

0 1

$$Det_{sg} \rightarrow \mathbf{some}$$

$$NBar \rightarrow \mathbf{natural \ number}$$

$$VP_{sg} \rightarrow \mathbf{is} \quad AP_{sg}$$

0

$$AP_{sg} \rightarrow \mathbf{prime}$$

Parsing a Sentence

- ▶ Consider all possible parse trees for the sentence.
- ▶ Apply filters to the collection of parse trees.
- ▶ How many parse trees are left?
 - ▶ 0 \rightarrow Ungrammatical
 - ▶ 1 \rightarrow Acceptable
 - ▶ 2+ \rightarrow Ambiguous
- ▶ Ungrammaticality and ambiguity are errors.

Parsing and Type

- ▶ Notation, terminology are (heavily) overloaded by type.

$$(A + B) = C \quad \text{vs} \quad \lambda + (\mathbf{M} = \mathbf{N})$$

- ▶ So we use types *during* parsing.

$$(A_{\text{NUMBER}} + B_{\text{NUMBER}}) = C_{\text{NUMBER}} \quad \text{OK}$$

$$A_{\text{NUMBER}} + (B_{\text{NUMBER}} = C_{\text{NUMBER}}) \quad \text{Not OK}$$

-
- ▶ Types can be assigned and consumed within one sentence.

Suppose that $K + f = x \in \mathcal{S}$, where K is an integer and $f \in [0, 1)$.

- ▶ So we can't know types without parsing.

Parsing with Types (Specification)

- ▶ Consider all possible parse trees.
- ▶ Use a filter on parse trees to select those where type assignments match type requirements.
- ▶ How should we define the filter?
- ▶ Types are semantic, i.e. *being a matrix* is a semantic property.
- ▶ So we need to work via the semantics.

- ▶ DRT-based compositional semantics.
- ▶ Semantic content attached to each rule.
- ▶ **All** semantic information is encoded in rules;
- ▶ This is needed to support adaptivity (growth via definitions).

Discourse Representation Theory (DRT)

- ▶ A logic for language. (Kamp and Reyle, 1993).
- ▶ *Discourse referents* instead of variables (Karttunen, 1996).
- ▶ Referents capture 'available entities' for anaphora.

| |
|---------------------------------------|
| x y |
| $John(x)$ $dog(y)$ $owns(x, y)$ |

John owns a dog.

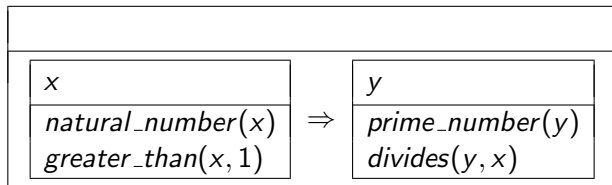
- ▶ Combine with Montague grammar.

$$\lambda x. group(x) \dashrightarrow \lambda x. \begin{array}{|c|} \hline \\ \hline group(x) \\ \hline \end{array}$$

Every natural number which is greater than 1 has a prime divisor.

If $n > 1$ is a natural number, then there is a prime p such that $p|n$.

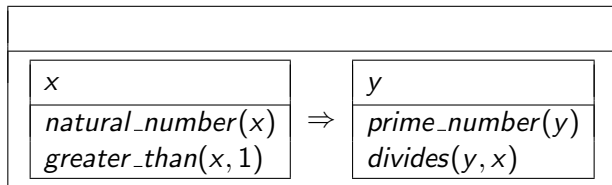
Variables as DRS Referents



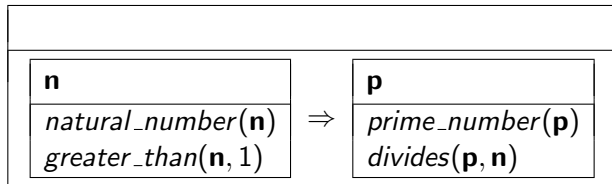
Every natural number which is greater than 1 has a prime divisor.

If $n > 1$ is a natural number, then there is a prime p such that $p|n$.

Variables as DRS Referents



Every natural number which is greater than 1 has a prime divisor.



If $n > 1$ is a natural number, then there is a prime p such that $p|n$.

Semantics (cont.)

$$S \rightarrow NP_{sg} \quad VP_{sg} \quad \{0(1)\}$$

0
 1

$$NP_{sg} \rightarrow Det_{sg} \quad NBar \quad \{0(1)\}$$

0
 1

$$Det_{sg} \rightarrow \mathbf{some} \quad \left\{ \lambda n. \lambda v. \left(\begin{array}{|c|} \hline r \\ \hline \end{array} \uplus n(r) \uplus v(r) \right) \right\}$$

$:e \rightarrow t$
 $:e \rightarrow t$

$$NBar \rightarrow \mathbf{natural\ number} \quad \left\{ \lambda x. \frac{}{natural_number(x:NUMBER)} \right\}$$

$:e$

$$VP_{sg} \rightarrow \mathbf{is} \quad AP_{sg} \quad \{0\}$$

0

$$AP_{sg} \rightarrow \mathbf{prime} \quad \left\{ \lambda x. \frac{}{prime_of_number(x_{NUMBER})} \right\}$$

$:e$

Ambiguity

- ▶ One word may have two or more meanings.
 - ▶ ‘There are infinitely many **prime** numbers.’
 - ▶ ‘Every nonzero **prime** ideal is maximal.’
- ▶ Each meaning corresponds to a separate entry in the grammar.

$$AP_{sg} \rightarrow \mathbf{prime} \left\{ \lambda_{X.} \frac{\quad}{:e} \boxed{\text{prime_ofnumber}(x_{\text{NUMBER}})} \right\}$$

$$AP_{sg} \rightarrow \mathbf{prime} \left\{ \lambda_{X.} \frac{\quad}{:e} \boxed{\text{prime_ofideal}(x_{\text{IDEAL}})} \right\}$$

- ▶ The correct entry must be determined during parsing.

Semantic Reduction

$$VP_{sg} \rightarrow \mathbf{is} \ AP_{sg} \quad \{0\}$$

$$AP_{sg} \rightarrow \mathbf{prime} \quad \left\{ \lambda x. \frac{\quad}{\text{prime_ofnumber}(x_{\text{NUMBER}})} \right\}$$

$$VP_{sg} \rightarrow \mathbf{is prime} \quad \left\{ \lambda x. \frac{\quad}{\text{prime_ofnumber}(x_{\text{NUMBER}})} \right\}$$

Semantic Reduction (cont.)

$$NP_{sg} \rightarrow Det_{sg} \underset{0}{NBar} \underset{1}{NBar} \quad \{0(1)\}$$

$$Det_{sg} \rightarrow \mathbf{some} \left\{ \lambda_{:e \rightarrow t} n . \lambda_{:e \rightarrow t} v . \left(\begin{array}{|c|} \hline r \\ \hline \end{array} \uplus n(r) \uplus v(r) \right) \right\}$$

$$NBar \rightarrow \mathbf{natural\ number} \left\{ \lambda_{:e} x . \begin{array}{|l|} \hline \\ \hline \text{natural_number}(x:\text{NUMBER}) \\ \hline \end{array} \right\}$$

$$NP_{sg} \rightarrow \mathbf{some\ natural\ number}$$

$$\left\{ \lambda_{:e \rightarrow t} v . \left(\begin{array}{|l|} \hline r \\ \hline \text{natural_number}(r:\text{NUMBER}) \\ \hline \end{array} \uplus v(r) \right) \right\}$$

- ▶ Actual type system contains parametric types, disjunctive types, nonextensional types, etc. . Not discussed.
- ▶ Types are semantic — so types must be computed on DRSs.

| |
|--------------------------------------------------------------|
| r |
| $natural_number(r:NUMBER)$ $prime_ofnumber(r_{NUMBER})$ |

- ▶ $natural_number(r:NUMBER)$ ← Type assignment.
 - ▶ $prime_ofnumber(r_{NUMBER})$ ← Type requirement.
 - ▶ $prime_ofideal(r_{IDEAL})$ ← Type requirement.
- ▶ **Idea behind type filter:** each type requirement must have a corresponding type assignment.

Types and Variables : Some Caveats

- ▶ Two variables can share the same name:

Then $\underbrace{f(x) > 0 \text{ for } x > 0}_{\text{First Binding Domain}}$ and $\underbrace{f(x) < 0 \text{ for } x < 0}_{\text{Second Binding Domain}}$.

- ▶ One variable can have different types in different parts of a single sentence.
- ▶ So we need to match up type assignments and type requirements carefully.

Types and Variables : Some Caveats

- ▶ Two variables can share the same name:

Then $\underbrace{f(x) > 0 \text{ for } x > 0}_{\text{First Binding Domain}}$ and $\underbrace{f(y) < 0 \text{ for } y < 0}_{\text{Second Binding Domain}}$.

- ▶ One variable can have different types in different parts of a single sentence.
- ▶ So we need to match up type assignments and type requirements carefully.

- ▶ 'Command' is a binary relation governing information flow
- ▶ Informally: A commands B if, when we are parsing B, we can rely on A being true.

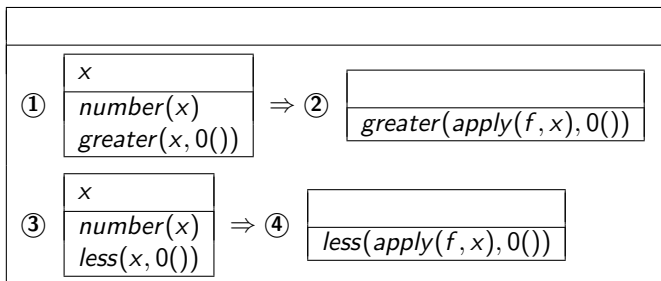
Then $\underbrace{f(x) > 0}_{\textcircled{1}}$ for $\underbrace{x > 0}_{\textcircled{2}}$ and $\underbrace{f(x) < 0}_{\textcircled{3}}$ for $\underbrace{x < 0}_{\textcircled{4}}$

Here $\textcircled{2}$ commands $\textcircled{1}$ and $\textcircled{4}$ commands $\textcircled{3}$.

- ▶ Since we are treating type on a semantic level, we need a semantic version of command.

Command (cont.)

- ▶ Semantic version of command:



(② commands ① and ④ commands ③ .)

- ▶ **Type Flow:** Each type requirement must be commanded by a corresponding assignment.

- ▶ Type requirements satisfied:

$S \rightarrow$ **some natural number is prime**

$$\left\{ \begin{array}{l} r \\ \textcircled{1} \textit{natural_number}(r:\text{NUMBER}) \\ \textcircled{2} \textit{prime_ofnumber}(r_{\text{NUMBER}}) \end{array} \right\}$$

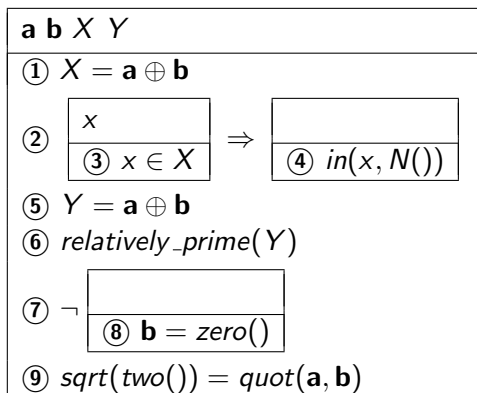
- ▶ Type requirements not satisfied:

$S \rightarrow$ **some natural number is prime**

$$\left\{ \begin{array}{l} r \\ \textcircled{1} \textit{natural_number}(r:\text{NUMBER}) \\ \textcircled{2} \textit{prime_ofideal}(r_{\text{IDEAL}}) \end{array} \right\}$$

($\textcircled{1}$ commands $\textcircled{2}$ in each example.)

Type Flow — A Real Example



① commands ② – ⑨; ② commands ⑤ – ⑨; ③ commands ④; ④ commands nothing; ⑤ commands ⑥ – ⑨; ⑥ commands ⑦ – ⑨; ⑦ commands ⑨; ⑧ commands nothing; ⑨ commands nothing.

Parsing with Types (recap.)

- ▶ Specification:
 - ▶ Consider all possible parse trees for the sentence.
 - ▶ Compositionally construct a DRS for each.
 - ▶ Track type flow around each DRS.
 - ▶ Discard DRS (and corresponding parse trees) where type constraints are not met.
- ▶ Doing this efficiently is another matter!

- ▶ Definitions change the grammar.
- ▶ Even definitions are part of the grammar.
- ▶ Example grammar rule capturing definitions:

$$\text{Def} \rightarrow \mathbf{a} \underbrace{\text{new-NBar}}_0 \mathbf{is} \mathbf{a} \underbrace{\text{NBar}_{sg}}_1 \left\{ \begin{array}{c} x \\ \hline 0(x) \stackrel{df}{\Leftrightarrow} 1(x) \end{array} \right\}$$

- ▶ Example definition:

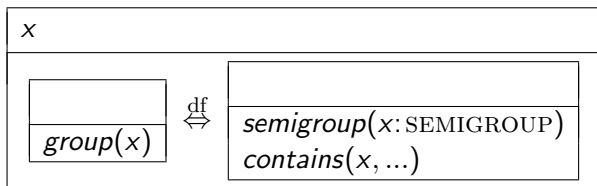
Definition. A group is a semigroup G containing

Adaptivity (cont.)

- ▶ Accepting the definition adds a new rule to the grammar...

$$NBar \rightarrow \mathbf{group} \left\{ \lambda x. \frac{\boxed{}}{\boxed{group(x)}} \right\}$$

- ▶ ... and the definition itself contributes semantic information...



- ▶ ... which (monotonically) adds type information to $group(\bullet)$:
 $group(x) \dashrightarrow group(x: SEMIGROUP)$

A Language for Mathematics

Mohan Ganesalingam

University of Cambridge, Computer Laboratory
mg262 at cam.ac.uk

March 10, 2009