Towards Multilingual Autoformalization and Informalization of Mathematics

Aarne Ranta
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
aarne.ranta@cse.gu.se

Abstract

Recent advances of AI in mathematics, such as AlphaProof's solutions of Mathematics Olympiad problems, combine large language models, which suggest solutions, with formal proof systems, which verify them. Conversion between formal and informal mathematical language is an essential part of this process. In AlphaProof, it is performed manually. This paper addresses the problem and reports preliminary results, building an automatic formalization system by combining grammar rules and data-driven parsing, and using Wikidata as a source of mathematical terminology.

1 Introduction

Mathematics is getting increasing attention in Artificial Intelligence. While undoubtedly an essential part of humans intelligence, mathematical reasoning is difficult for purely data-driven methods such as neural networks (Roberts, 2023). The problem is precision and reliability: just like a chess program is useless if it does not obey the rules of chess, mathematical reasoning is useless if it does not obey the rules of mathematics.

A solution is offered by **formal mathematics**, systems that express mathematical reasoning in a formal language where correctness can be checked mechanically. A way to combine such systems with, for instance, large language models (LLMs), is to let the LLM generate solutions in a formal language and the formal system check them until a correct solution is found. One such system, AlphaProof, was recently used for solving problems in the Mathematics Olympiad (AlphaProof and AlphaGeometry teams, 2024). In AlphaProof, a pre-trained LLM interacts with the formal proof system Lean (de Moura et al., 2015).

Solving a problem by AlphaProof starts by translating natural language problem statements into the formal language used in Lean. This step is performed manually, which is a drawback. Another drawback is the lack of training material: while most existing mathematics is written in natural languages (mixed with mathematical symbols), using these texts without a precise connection to formalized mathematics does not guarantee correctness.

In this paper, we will outline a solution to the problem and present some preliminary results. Translation from informal to formal is known as **autoformalization**. The opposite direction can be called **informalization**. In the emerging project Informath, Informalization of Formal Mathematics, we are looking at both directions, by treating autoformalization as the inversion of informalization. In this paper, we will summarize an experimental system aimed to translate a corpus of real-world mathematical texts from English to Lean as well as to some other formal and natural languages.

2 Autoformalization and informalization

The topic of translation between formal and informal mathematics is not new (de Bruijn, 1994; Ranta, 1994; Coscoy et al., 1995). The original goal was to enable formal proof systems to be accessed via natural language. Early approaches used rule-based methods such as formal grammars, defining controlled natural languages (CNLs).

The term autoformalization came later, in connection to data-driven AI methods. A pioneering work is Wang et al. (2020), which uses neural machine translation. A recent approach, GFLean (Pathak, 2024), uses rule-based methods again: the Grammatical Framework (GF, Ranta, 2011; Ranta et al., 2020) in combination with formal semantics of Montague style (Montague, 1974).

GFLean uses GF for parsing ForTheL (Paskevich, 2007), an English-like CNL for mathematics. The resulting syntax trees are converted to logical representations, from which Lean code is produced by **linearization** (converting trees to strings) by GF. Figure 1 shows the architecture of GFLean and an example formalization.

The Informath project reported in this paper extends GFLean in the following ways:

- The syntax of ForTheL is extended to cover more constructs, guided by existing mathematical texts.
- The grammar is made multilingual, so that it addresses multiple natural languages instead of just English.
- The system supports both formalization and informalization.
- The syntax is combined with an extensive lexicon of mathematical terms.

The architecture of Informath is shown in Figure 1 (c). It can be seen as a magnified picture of Figure 1 (a), where the English CNL is generalized to a multilingual natural language component, and the target language Lean is generalized to an abstract logical representation, which can be related to other systems of formal mathematics as well.

Informath aims to grow beyond CNL and approach arbitrary mathematical text. As a first step, we have extended the ForTheL syntax so that we can parse the original statements from Chartrand et al. (2007) instead of manually translating them to ForTheL. For example, to cover Figure 1 (b), we extended the syntax of English with noun phrase coordination and post-quantification and the syntax of mathematical symbolism with operatorless multiplication. Via the abstract syntax, the parse tree can be converted to Lean by the same semantic rules as in original GFLean. The ForTheL CNL thus plays the role of the "core abstract syntax" of Figure 1 (c).

3 Mathematical syntax and terminology

The grammar of natural language in Informath uses the GF Resource Grammar Library (RGL, Ranta, 2009), which provides functions for the morphology and surface syntax of over 40 languages. It can be used via a **functor**, where the shared syntactic structure of the RGL are combined with language-specific morphological rules and lexical entries. To add a new language, just a lexical rule needs to be added. For example, the predicate "x is even" need a translation of the adjective *even*, whereas the syntactic rule for adjectival predication is uniformly defined by the functor. The resulting grammar can both parse and generate the predicate in different comninations and forms, such as *om x och y är jämna, så är summan av x och y jämn* (Swedish for *if x and y are even, then the sum of x and y is even*).

The syntax of mathematical texts, with its mixture of natural language of symbols, is a challenge in itself (Ganesalingam, 2013). But it can be largely covered by a few hundred GF functions; the extension of ForTheL reported here has around 100 abstract syntax functions. The bulk of mathematical language is **terminology**: the precise expressions for mathematical concepts such as sets, functions, and algebraic structures. In this respect, mathematics is similar to other domains of knowledge, such as medicine or law.

Terminology is also in focus when teaching and documenting knowledge. For instance, a typical Wikipedia article has a technical term as its title, and the articles in different languages have corresponding terms of those languages as their titles. These correspondences have been encoded in Wikidata (Vrandečić and Krötzsch, 2014), where every concept has a unique identifier and a set of **labels** in different languages.

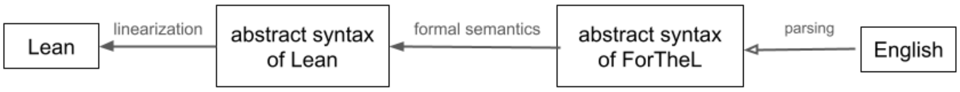
In mathematics, a typical use of a term is as the definiens of a definition, such as *Abelian group* in Figure 2. Definitions are typically stated at the beginning of Wikipedia articles. One of the outcomes of Informath, in addition to autoformalization, is to contribute to the Abstract Wikipedia project, which generates articles from formal representations (Vrandečić, 2021; Ranta, 2023). A larger set of examples is given in Appendix C.

4 Terminology extraction

We have developed a conversion from Wikidata labels to grammar rules in GF. These rules have to implement both morphology and syntax. Morphology specifies the inflection words, as well as their properties such as part of speech and gender. Syntax specifies the structure needed for the correct use of these features. For example, the term *groupe abélien* in French is a combination of a masculine noun and an adjective. Both gender and number agreement are needed: the plural is *groupes abéliens*. Our conversion enriches the raw-text Wikidata labels with enough information to determine all this behaviour.

The conversion is a combination of GF and Universal Dependencies (UD, (de Marneffe et al., 2021)). It covers all steps from Wikidata to GF grammars: it first parses the labels with UD to extract lemmas and parts of speech and extends the standard GF grammar with new entries obtained in

(a)



(b)

Theorem 14 (Exercise 3.10)**.** *If a and c are odd integers, then $ab+ac$ is even for every integer b.*

```
Ex. Assume a is an odd integer and c is an odd integer. Then for every integer b,
a * b + a * c is even.
```

```
example (a : ℤ) (h78 : odd a) (c : ℤ) (h57 : odd c) :
∀ (b : ℤ), even ((a * b) + (a * c)) := sorry
```

(c)

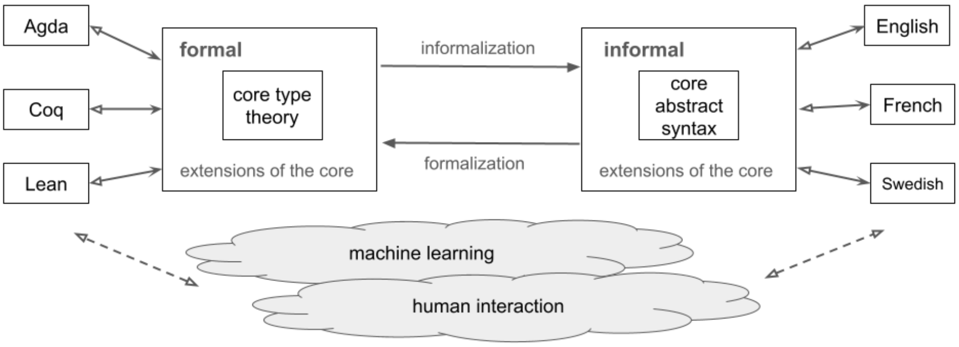


Figure 1: (a) The system GFLean translating the ForTheL fragment of English to the formal proof system Lean. Solid arrowheads indicated total functions, hollow arrowheads partial functions.
(b) An example of GFLean, formalizing a statement from a textbook (Chartrand et al., 2007). The first line is the original statement. The second line is its manual conversion to ForTheL. The rest is the automatically generated formalization in Lean. (Notice that there is a typo in the statement: it should say $ab+bc$.)
(c) The big picture of the Informath project.

this process. Then it parses the labels with with the extended GF grammar and generates GF modules for the Wikidata terms. More details of this process are given in Appendix B.

5 Evaluation

Coverage. Table 1 shows the number of extracted terms from 5381 Wikidata labels in 8 languages. These labels cover the 683 concepts of an undegraduate mathematics curriculum in the MathGloss project (Horowitz and de Paiva, 2023), extended with various SPARQL searches in Wikidata. The coverage of labels varies greatly among languages. This can only be improved by extending the coverage of Wikidata itself. Even if the labels are given, the current GF parser is not yet complete. This can be improved by extending the GF grammar and by making better use of the UD parse trees with methods described in Ranta et al. (2020).

language	labels covered		successful parses	
Eng	5188	96%	3872	74%
Fin	834	15%	328	39%
Fre	3230	60%	2199	68%
Ger	2956	54%	2609	88%
Ita	2019	37%	1390	68%
Por	2858	53%	1717	60%
Spa	2322	43%	1633	70%
Swe	1345	24%	826	61%

Table 1: The coverage of Wikidata labels and their successful GF parses for different languages. The total amount of Wikidata items is 5381.

The coverage of the syntax part is an extension of the ForTheL fragment of Pathak (2024) and manages to parse a majority of the original statements from Chartrand et al. (2007). An obvious way to

English: An Abelian group is a group whose binary operation is commutative.
Abstract syntax: `ParDefinition (DefWhose (KindQN abelian_group_Q181296_QN)`
`(KindQN group_Q83478_QN) (KindQN binary_operation_Q164307_QN) commutative_Property)`
Lean: `def Abelian_group := {x : Group // (Commutative (binary_operation x))}`
Finnish: Abelin ryhmä on ryhmä, jonka binäärioperaatio on kommutatiivinen.
French: Un groupe abélien est un groupe dont l'opération binaire est commutative.
German: Eine abelsche Gruppe ist eine Gruppe, deren zweistellige Verknüpfung kommutativ ist.
Italian: Un gruppo abeliano è un gruppo di cui operazione binaria è commutativa.
Swedish: En abelsk grupp är en grupp vars binära operator är kommutativ.

Figure 2: An autoformalization and some translations of a definition, obtained by parsing the English statement and linearizing it to the other formats. The abstract syntax segments of the form "Q"* are Wikidata identifiers, and their linearizations in different languages have been derived from Wikidata labels.

extend it is to extend the GF grammar; however, also a more robust data-driven method could be considered (Ranta et al., 2020).

Quality. The accuracy of the extracted terms has only been evaluated by an ocular inspection of examples. It is dependent on the quality of the Wikidata labels, which, in the mathematics domain, seems to be relatively high. It also depends on the quality of the GF grammars used for extraction; in particular, the morphological properties of new words that only appear once in the material are just educated guesses using GF's "smart paradigms" (Détrez and Ranta, 2012).

In syntax, the main challenge is ambiguity: autoformalization with GF grammars, even with the relatively small one we have now, may result in numerous parse trees. In a thorough analysis of the language of mathematics, Ganesalingam (2013) suggests that syntactic ambiguity is unavoidable, but that it is always resolved by semantic clues. Piping syntactic parses into a formal proof system provides one way to do this.

Workload. A common worry about rule-based methods is that they require manual work. However, if high precision is needed, as in mathematics, even the best data-driven methods require manual work in checking the machine-generated output. Then it makes sense to work on rules that can be improved so that the need of manual checking continuously decreases.

The best of both worlds is to let data-driven methods produce rules, which a human can improve without having to produce them from scratch. This is what we do in the derivation of lexical rules from Wikidata labels using UD parsing. The programming effort for the reported system (by the author) was two weeks. Each new language was added in a couple of hours and just 10% language-specific rules (see Appendix A). This was of course only possible thanks to the existing RGL, where adding a language typically requires a few months of work. But this work is amortized as the RGL is used in new applications.

6 Conclusion

We have shown some early results from the project Informath, Informalization of Formal Mathematics. The project applies GF grammars to target multiple natural and formal languages. The experiment reported here had as its starting point the controlled language ForThel. as used in the project GFLean, which translates a fragment of English to the proof system Lean. We extended the grammar to autoformalize the original textbook statements used in GFLean without manual conversion. We also generalized it to eight natural languages, using Wikidata labels as the source of a multilingual terminology. The goal of Informath is to provide a reliable resource for translating between formal and informal mathematics. It aims to help users of proof systems, Wikipedia users wanting to read about mathematics in their own languages, and AI systems that need to convert between formal and informal languages when solving mathematical problems.

Acknowledgements

Much of the work in this paper was done during a four-week stay at Hausdorff Institute for Mathematics in Bonn during the trimester "Prospects of Formal Mathematics" in July 2024. I am grateful to the institute and the organizers for the invitation and financial support. The work would not have been possible without the exchange of ideas with other participants. Lucy Horowitz, Jan Frederik Schaefer, Peter Koepke, and Shashank Pathak provided the material used for building the experiment and evaluating it. Hans Leiß, Josef Urban, Marcel Schütz, Mario Carneiro, Michael Kohlhase, Paul-André Melliès, and Valeria de Paiva helped with their expertise, ideas, and challenging questions.

References

AlphaProof and AlphaGeometry teams. 2024. AI achieves silver-medal standard solving International Mathematical Olympiad problems.

G. Chartrand, A. D. Polimeni, and P. Zhang. 2007. *Mathematical Proofs*. Pearson.

Y. Coscoy, G. Kahn, and L. Thery. 1995. Extracting text from proofs. In *Proc. Second Int. Conf. on Typed Lambda Calculi and Applications*, volume 902 of *LNCS*, pages 109–123.

Marcos Cramer, Bernhard Fisseni, Peter Koepke, Daniel Kühlwein, Bernhard Schröder, and Jip Veldman. 2009. The Naproche Project Controlled Natural Language Proof Checking of Mathematical Texts. In *CNL*, volume 5972 of *LNCS*, pages 170–186. Springer.

N. G. de Bruijn. 1994. Mathematical Vernacular: a Language for Mathematics with Typed Sets. In R. Nederpelt, editor, *Selected Papers on Automath*, pages 865–935. North-Holland Publishing Company.

Marie-Catherine de Marneffe, Christopher D. Manning, Joakim Nivre, and Daniel Zeman. 2021. Universal Dependencies. *Computational Linguistics*, 47(2):255–308.

Leonardo de Moura, Soonho Kong, Jeremy Avigad, Floris van Doorn, and Jakob von Raumer. 2015. The lean theorem prover (system description). In *Automated Deduction - CADE-25*, pages 378–388, Cham. Springer International Publishing.

G. Détrez and A. Ranta. 2012. Smart paradigms and the predictability and complexity of inflectional morphology. In *EACL 2012*.

Mohan Ganesalingam. 2013. *The Language of Mathematics: A Linguistic and Philosophical Investigation*. Springer.

Lucy Horowitz and Valeria de Paiva. 2023. Mathgloss: Building mathematical glossaries from text. *Preprint*, arXiv:2311.12649.

R. Montague. 1974. *Formal Philosophy*. Yale University Press, New Haven. Collected papers edited by Richmond Thomason.

Andrei Paskevich. 2007. The Syntax and Semantics of The ForTheL Language.

Shashank Pathak. 2024. GFLean: An Autoformalisation Framework for Lean via GF. *Preprint*, arXiv:2404.01234.

L. Paulson. 2002. The Isabelle Reference Manual. Available at the Isabelle homepage. With contributions by T. Nipkow and M. Wenzel.

Aarne Ranta. 1994. Type theory and the informal language of mathematics. In *Selected papers from TYPES'93: Int. Workshop on Types, Nijmegen, The Netherlands*, volume 806 of *LNCS*, pages 352–365. Springer-Verlag.

Aarne Ranta. 2009. The GF Resource Grammar Library. *Linguistics in Language Technology*, 2.

Aarne Ranta. 2011. *Grammatical Framework: Programming with Multilingual Grammars*. CSLI Publications, Stanford.

Aarne Ranta. 2023. Multilingual text generation for abstract wikipedia in grammatical framework: Prospects and challenges. In *Logic and Algorithms in Computational Linguistics 2021 (LACompLing2021)*, pages 125–149, Cham. Springer Nature Switzerland.

Aarne Ranta, Krasimir Angelov, Normunds Gruzitis, and Prasanth Kolachina. 2020. Abstract Syntax as Interlingua: Scaling Up the Grammatical Framework from Controlled Languages to Robust Pipelines. *Computational Linguistics*, 46(2):425–486.

Siobhan Roberts. 2023. A.I. Is Coming for Mathematics, Too. The New York Times.

Denny Vrandečić. 2021. Building a Multilingual Wikipedia. *Communications of the ACM*, 64(4):38–41.

Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Qingxiang Wang, Chad Brown, Cezary Kaliszyk, and Josef Urban. 2020. Exploration of neural machine translation in autoformalization of mathematics in mizar. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2020, page 85–98, New York, NY, USA. Association for Computing Machinery.

A The grammars

The experiment reported in this paper involves two grammars: one for lexicon extraction and another for the runtime, i.e., the actual formalization and informalization if mathematical texts.

The extraction grammar parses Wikipedia labels and finds their syntactic structure. Terms are typically multiword expressions. When used in different contexts, it is essential to know their syntactic structure, to know the behaviour of each word in them. The results of lexicon extraction are collected to a multilingual lexicon of mathematical terms, where the abstract syntax functions are named after Wikidata identifiers, to which English labels are added for readability. Some examples are shown in Figure 2. The size of the extracted lexicon is shown in Table 1, where the "successful parse" column indicates the coverage of the extraction grammar.

The grammar used at runtime combines the extracted lexicon with a syntax part, which is an extension of ForTheL. The syntax part has at the time of writing 78 combination functions. In addition, there are 57 functions for mathematical symbolism, which is linearized in the same way in all languages.

Following a usual pattern in GF, the syntax part is implemented via a functor, which maps the semantic structures of the application to the abstract syntactic structures of the RGL. The RGL implements these structures separately for each individual language, so that in an application such as Informath, no language-specific code needs to be written. However, languages might still use different syntactic structures for particular semantic structures. To deal with this, a functor can in each language have **exceptions**, which override the shared implementation. For example, the universal quantifier phrase *all functions* resulting from the functor is in French (and some other languages) overridden by a rule that inserts a definite article: *toutes les fonctions*.

Due to functors, both the extraction and the formalization grammar require very few hand-written rules. By counting the number of exceptions and additions to the functor, we can estimate the need of manual work in them. The following table gives the number of rules explicitly written in each parts of the grammar. For "functor", this means the number of language-independent definitions in terms of RGL functions. For each individual language, this means the number of exceptions to the functor. The numbers imply that around 90% of rules in each language are shared in both tasks, and have therefore not required any manual work.

language	Extract	ForTheL
functor	21	133
Eng	2	11
Fin	2	14
Fre	3	16
Ger	2	21
Ita	3	15
Por	3	15
Spa	3	17
Swe	2	20

B The lexicon extraction process

The following run of the lexicon extraction script (written in Python) adds German to the grammar. On a Macbook Air M2 2023, it runs in less than 30 seconds. Most messages have been omitted, but the most relevant statistics output is left.

```
$ time ./build_lexicon.py -add de Ger
* Step 1: Extracting Wikidata labels
  statistics terms: 5381
  statistics no label: 2425
* Step 2: Parsing data with UDPipe
* Step 3: Analysing data with UD results
  statistics case fixed: 207
  statistics words from UD: 2691
* Step 4: Building lexicon extension
  statistics given morphofuns: 44227
  statistics new extracted morphofuns: 2472
* Step 5: Parsing with GF
  statistics successful GF parses: 2527
  statistics failed GF parses: 429
* Step 7: add a new concrete syntax
* Step 8: testing the grammar in GF
  MathTerms: cubic_graph_Q1374495_CN
  MathTermsGer: s Strong Sg Nom : kubischer Graph
  MathTermsGer: s Strong Sg Acc : kubischen Graph
  MathTermsGer: s Strong Sg Dat : kubischem Graph
  MathTermsGer: s Strong Sg Gen : kubischen Graphs
  MathTermsGer: s Strong Pl Nom : kubische Graphe
  MathTermsGer: s Strong Pl Acc : kubische Graphe
  MathTermsGer: s Strong Pl Dat : kubischen Graphen
  MathTermsGer: s Strong Pl Gen : kubischer Graphe
  MathTermsGer: s Weak Sg Nom : kubische Graph
  # ... plus 15 more forms of common noun phrases
13.74s user 0.99s system 52% cpu 28.097 total
```

The code, both for grammars and Python scripts, is available in `https://github.com/aarneranta/gf-math`.

C Examples from "100 theorems"

The following ten subsections are a sample of the "100 greatest theorems", which have been used as a a benchmark for formal systems of mathematics[1]. We show it here instead of the GFLean corpus, because it is an independent set of examples, for which the grammar was not originally designed. It also has more terminological variation and is thereby a better illustration of the extracted terminology.

This sample has been formalized in the Naproche system (Cramer et al., 2009), which uses a version of ForTheL combined with LaTeX as its input language and translates it to the Isabelle proof system (Paulson, 2002). In our experiment, we parsed the English theorem statements with the Informath grammar. The resulting abstract syntax was linearized to eight languages: English (identical to the input), Finnish, French, German, Italian, Portuguese, Spanish, and Swedish, which in each subsection are shown in this order. Since the grammars are reversible, each of the languages could equally well be used as an input of formalization in Isabelle. The text below is the unedited output of the system, and the readers are invited to spot errors in their own languages.

---

[1] `https://www.cs.ru.nl/~freek/100/`

1

$q^2 = p$ for no positive rational number q.
$q^2 = p$ millekään positiiviselle luvulle q.
$q^2 = p$ pour aucun nombre rationnel positif q.
$q^2 = p$ für keine positive rationale Zahl q.
$q^2 = p$ per nessuno numero razionale positivo q.
$q^2 = p$ para nenhum número racional positivo q.
$q^2 = p$ para ningún número racional positivo q.
$q^2 = p$ för inget positivt rationellt tal q.

2

The collection of prime numbers is infinite.
Kokoelma alkulukuja on ääretön.
La collection de nombres premiers est infinie.
Die Gesamtheit von Primzahlen ist unendlich.
La collezione di numeri primi è infinita.
O collection de números primos é infinito.
La colección de números primos es infinita.
Samlingen av primtal är oändlig.

3

Let x, y be sets. x and y are equinumerous iff there exists a injective map from x to y and there exists an injective map from y to x.
Olkoot x, y joukkoja. x ja y ovat yhtämahtavia jos ja vain jos on olemassa injektiivinen kuvaus x:sta y:an ja on olemassa moduli kuvaus y:sta x:an.
Soient x, y des ensembles. x et y sont équinombreux si et seulement si il existe une correspondance injective de x à y et il existe une application injective de y à x.
Seien x, y Mengen. x und y sind gleichzahlig wenn und genau dann wenn es eine injektive Abbildung aus x nach y gibt und es eine injektive Abbildung aus y nach x gibt.
Siano insiemi x, y. x e y sono equinumerosi se e solo se esiste una mappa iniettiva da x a y ed esiste una mappa iniettiva da y a x.
Deixe x, y ser conjuntos. x e y são equinumeiros se e só se existe uma função injetiva de x a y e existe uma aplicação injetiva de y a x.
Supongamosnos que x, y son conjuntos. x y y son equinumerosos si y solo si existe una función inyectiva de x a y y existe una función inyectiva de y a x.
Låt x, y vara mängder. x och y är liktaliga om och endast om det finns en injektiv avbildning från x till y och det finns en injektiv avbildning från y till x.

4

For all finite sets X and all natural numbers n, if $|X| = n$, then $\mathscr{P}(X)$ is finite and $|\mathscr{P}(X)| = 2^n$.
Kaikille äärellisille joukoille X ja kaikille luonnollisille luvuille n, jos $|X| = n$, niin $\mathscr{P}(X)$ on äärellinen ja $|\mathscr{P}(X)| = 2^n$.
Pour tous les ensembles finis X et tous les entiers naturels n, si $|X| = n$, alors $\mathscr{P}(X)$ est fini et $|\mathscr{P}(X)| = 2^n$.
Für alle endlichen Mengen X und alle natürlichen Zahlen n, wenn $|X| = n$, dann ist $\mathscr{P}(X)$ endlich und $|\mathscr{P}(X)| = 2^n$.
Per tutti gli insiemi finiti X e tutti i numeri naturali n, se $|X| = n$, allora $\mathscr{P}(X)$ è finito e $|\mathscr{P}(X)| = 2^n$.
Para todos os conjuntos finitos X e todos os números naturais n, se $|X| = n$, então $\mathscr{P}(X)$ é finito e $|\mathscr{P}(X)| = 2^n$.
Para todos los conjuntos finitos X y todos los números naturales n, si $|X| = n$, entonces $\mathscr{P}(X)$ es finito y $|\mathscr{P}(X)| = 2^n$.
För alla ändliga mängder X och alla naturliga tal n, om $|X| = n$, så är ändligt $\mathscr{P}(X)$ och $|\mathscr{P}(X)| = 2^n$.

5

Let s,t be real numbers such that $s < t$. Then there exists a real number r such that $s < r < t$.
Olkoot s,t reaalilukuja siten että $s < t$. Silloin on olemassa reaaliluku r siten että $s < r < t$.
Soient s,t des nombres tel que $s < t$. Alors il existe un nombre r tel que $s < r < t$.
Seien s,t reelle Zahlen derart dass $s < t$. Dann gibt es eine reelle Zahl r derart dass $s < r < t$.
Siano numeri tale che $s < t$ s,t. Allora esiste un numero r tale che $s < r < t$.
Deixe s,t ser números tal que $s < t$. Então existe um número r tal que $s < r < t$.
Supongamosnos que s,t son números tal que $s < t$. Entonces existe un número r tal que $s < r < t$.
Låt s,t vara tal så att $s < t$. Då finns det ett tal r så att $s < r < t$.

6

Let M be a set. Then there exists no surjection from M onto the powerset of M.
Olkoon M joukko. Silloin ei ole olemassa mitään surjektiota M:sta potenssijoukolle M:n.
Soit M un ensemble. Alors il n'existe aucune surjection de M sur l'ensemble puissance de M.
Sei M eine Menge. Dann gibt es keine Surjektion aus M auf die Potenzmenge M.

Sia un insieme M. Allora non esiste nessuna suriezione da M sull'insieme delle parti di M.

Deixe M ser um conjunto. Então não existe nenhuma sobrejecção de M sobre o conjunto de potência de M.

Supongamosnos que M es un conjunto. Entonces no existe ninguna sobreyección de M sobre el conjunto potencia de M.

Låt M vara en mängd. Då finns det ingen surjektion från M på potensmängden av M.

7

$\sum_{0\leq i<n} x^i = \frac{1-x^n}{1-x}$ for all natural numbers n.

$\sum_{0\leq i<n} x^i = \frac{1-x^n}{1-x}$ kaikille luonnollisille luvuille n.

$\sum_{0\leq i<n} x^i = \frac{1-x^n}{1-x}$ pour tous les entiers naturels n.

$\sum_{0\leq i<n} x^i = \frac{1-x^n}{1-x}$ für alle natürlichen Zahlen n.

$\sum_{0\leq i<n} x^i = \frac{1-x^n}{1-x}$ per tutti i numeri naturali n.

$\sum_{0\leq i<n} x^i = \frac{1-x^n}{1-x}$ para todos os números naturais n.

$\sum_{0\leq i<n} x^i = \frac{1-x^n}{1-x}$ para todos los números naturales n.

$\sum_{0\leq i<n} x^i = \frac{1-x^n}{1-x}$ för alla naturliga tal n.

8

$\sum_{i=1}^{n}(a+d\cdot i) = n\cdot(a+\frac{(n+1)\cdot d}{2})$..

$\sum_{i=1}^{n}(a+d\cdot i) = n\cdot(a+\frac{(n+1)\cdot d}{2})$..

$\sum_{i=1}^{n}(a+d\cdot i) = n\cdot(a+\frac{(n+1)\cdot d}{2})$..

$\sum_{i=1}^{n}(a+d\cdot i) = n\cdot(a+\frac{(n+1)\cdot d}{2})$..

$\sum_{i=1}^{n}(a+d\cdot i) = n\cdot(a+\frac{(n+1)\cdot d}{2})$..

$\sum_{i=1}^{n}(a+d\cdot i) = n\cdot(a+\frac{(n+1)\cdot d}{2})$..

$\sum_{i=1}^{n}(a+d\cdot i) = n\cdot(a+\frac{(n+1)\cdot d}{2})$..

$\sum_{i=1}^{n}(a+d\cdot i) = n\cdot(a+\frac{(n+1)\cdot d}{2})$..

9

Let m,n be natural numbers such that $m < n$. Then the greatest common divisor of m and n is the greatest common divisor of $n-m$ and m.

Olkoot m,n luonnollisia lukuja siten että $m < n$. Silloin suurin yhteinen tekijä m:n ja n:n on suurin yhteinen tekijä $n-m$:n ja m:n.

Soient m,n des entiers naturels tel que $m < n$. Alors le plus grand commun diviseur de m et de n est le plus grand commun diviseur de $n-m$ et de m.

Seien m,n natürliche Zahlen derart dass $m < n$. Dann ist der größte gemeinsame Teiler m und n der größte gemeinsame Teiler $n-m$ und m.

Siano numeri naturali tale che $m < n$ m,n. Allora il massimo comun divisore di m e n è il massimo comun divisore di $n-m$ e m.

Deixe m,n ser números naturais tal que $m < n$. Então o máximo divisor comum de m e n é o máximo divisor comum de $n-m$ e m.

Supongamosnos que m,n son números naturales tal que $m < n$. Entonces el máximo común divisor de m y n es el máximo común divisor de $n-m$ y m.

Låt m,n vara naturliga tal så att $m < n$. Då är den größta gemeinsama Teilern av $n-m$ och m den größta gemeinsama Teilern av m och n.

10

Assume $A \subseteq \mathbb{N}$ and $0 \in A$ and for all $n \in A$, $n+1 \in A$. Then $A = \mathbb{N}$.

Oleta, että $A \subseteq \mathbb{N}$ ja $0 \in A$ ja kaikelle $n \in A$:lle, $n+1 \in A$. Silloin $A = \mathbb{N}$.

Supposons que $A \subseteq \mathbb{N}$ et $0 \in A$ et pour tout $n \in A$, $n+1 \in A$. Alors $A = \mathbb{N}$.

Wir nehmen an, dass $A \subseteq \mathbb{N}$ und $0 \in A$ und für alle $n \in A$, $n+1 \in A$. Dann $A = \mathbb{N}$.

Supponiamo che $A \subseteq \mathbb{N}$ e $0 \in A$ e per tutto $n \in A$, $n+1 \in A$. Allora $A = \mathbb{N}$.

Admitemos que $A \subseteq \mathbb{N}$ e $0 \in A$ e para todo $n \in A$, $n+1 \in A$. Então $A = \mathbb{N}$.

Supongamosnos que $A \subseteq \mathbb{N}$ y $0 \in A$ y para todo $n \in A$, $n+1 \in A$. Entonces $A = \mathbb{N}$.

Vi antar att $A \subseteq \mathbb{N}$ och $0 \in A$ och för allt $n \in A$, $n+1 \in A$. Då $A = \mathbb{N}$.