Content-based encoding of mathematical and code libraries

Josef Urban
Institute for Computing and Information Sciences
Radboud University, Nijmegen

August 27, 2011

Overview

- Introduction: Formal math libraries and wikis
- Motivation: naming problems and their implications
- Content-based naming methods
- Proposed usage in math libraries
- Limitations and extensions
- Feedback is appreciated!

Introduction: Formal math libraries and wikis

- Mathematics can be expressed fully formally
- This allows detailed computer understanding
- Similar to code libraries
- Proof verification (analogous to code compilation) is then possible
- Strong computer assistance possible: automated reasoning, semantic search
- Large formal libraries arise, similar to code libraries: Mizar, Coq, Isabelle, HOL
- Some problems very similar to software libraries management
- Actually, we do not know a crisp boundary between code and formal math (Prolog is clearly both)

Motivation: naming problems and their implications

- Bolzano-Weierstrass theorem or just Weierstrass theorem?
- Solomonoff vs. Kolmogorov vs. Chaitin complexity vs. algorithmic entropy?
- In a formal library: relation_composition(R,S) or compose(R,S) or R*S ?
- many more (additive vs multiplicative groups, operations on all kinds of numbers ...)

Motivation: naming problems and their implications

- Renaming: Weierstrass gets renamed to Bolzano-Weierstrass
- Moving: CoRN.algebra.Basics.iterateN becomes CoRN.utilities.iterateN .
- Merging: Chaitin complexity and Kolmogorov complexity are found to be the same thing
- All these operations cause syntactic change of the depending proofs and theorems

Motivation: naming problems and their implications

- However, the changes are purely syntactic, there is no *semantic* difference
- How do we align two different concepts spaces with each other?
- How do we use various searching and automated reasoning tools *modulo* the different syntactic concept hierarchies?
- One use-case: a new user comes with his own vocabulary and does not know the concepts in a large library

Current naming methods

- serial numbering of theorems in textbooks and in Mizar: CARD_1:def 1
- module-based paths in Coq: CoRN.algebra.Basics.iterateN or CoRN.utilities.iterateN
- possibly somewhat more descriptive names: commutativity_of_plus
- name mangling: types of arguments added explicitly to the name
- none of these are strictly depending on the semantics (contents) of the items

Content-based naming methods

- Gödel numbering
- Recursive term sharing
- Recursive cryptographic hashing

Content-based naming methods: Gödel numbering

- basic logic objects are assigned natural numbers
- complicated objects are modelled from less complicated as sequences
- a one-to-one encoding of finite sequences to numbers
- thus, every mathematical object is uniquelly assigned (a very large) number based purely on its contents
- this gives us (theoretically) purely content-based indentifiers
- however, this does not seem to be practically usable, the numbers will be very large

Content-based naming methods: Recursive term sharing

- automated/interactive theorem provers (ATPs), Prolog
- exhaustive sharing of terms is used to achieve space/time efficiency
- example: `f(g(a)), g(g(a))` is represented as:
- `a -> *0, g(*0) -> *1, f(*1) -> *2, g(*1) -> *3`
- difference to Gödel numbering: objects are numbered serially as they come
- this makes this scheme fragile
- in some sense, not perfectly content-based, depending also on ordering

Content-based naming methods: Recursive cryptographic hashing

- Gödel numbering results in impractically large identifiers
- Recursive term sharing too fragile
- Is there something usable?
- Minimal perfect hashing? Not really feasible for math objects
- Cryptographic hashing! SHA1 SHA256 used in git
- Conflicts are extremely unlikely
- SHA1 results in 40-character identifiers - this is feasible!

Content-based naming of formal mathematics

- The initial library items get an SHA1 value (e.g. their SHA1 value as strings, etc.) that does not change between the library versions
- A suitable semantic form (XML) is defined for terms, formulas, etc.
- The SHA1 of the semantic form (tree, DAG of items - SHA1 values) is used as the content-based identifier
- This is very similar to the way how git recursively computes fie/directory names

Proposed use

- See how much naming-based duplication is inside the libraries
- Multiplicative vs. additive versions of algebraic structures
- Tracking the items' histories during wiki-like refactoring:
- Where were items moved, how were they renamed (semantic diff)
- Name-independent automated reasoning/search tools over the libraries:
- Should be useful particularly for new users that do not know the canonical concept names

Limitations and extensions

- Wikipedia article typically keeps its name for long time, even though its content changes
- This gives rise to an equivalence class of SHA1 hashes
- Such equivalence classes need to be propagated using some kind of congruence closure algorithm
- Semiformal libraries: take SHA1 only of the formal content (skip the comments)
- Interesting issue is normalization:
- Alternative versions of associative-commutative operations should be normalized into the same semantic form before the SHA1 is computed