Appendix to *Simple Dataset for Proof Method Recommendation in Isabelle/HOL (Dataset Description)* ⋆

Yutaka Nagashima[1,2][0000−0001−6693−5325]

[1] Czech Technical University in Prague, Prague, Czech Republic
Yutaka.Nagashima@cvut.cz
[2] University of Innsbruck, Innsbruck, Austria

Abstract. Recently, a growing number of researchers have applied machine learning to assist users of interactive theorem provers. However, the expressive nature of underlying logics and esoteric structures of proof documents impede machine learning practitioners, who often do not have much expertise in formal logic, let alone Isabelle/HOL, from achieving a large scale success in this field. In this data description, we present a simple dataset that contains data on over 400k proof method applications along with over 100 extracted features for each in a format that can be processed easily without any knowledge about formal logic. Our simple data format allows machine learning practitioners to try machine learning tools to predict proof methods in Isabelle/HOL without requiring domain expertise in logic.

1 Introduction

The 13th Conference on Intelligent Computer Mathematics (CICM 2020) accepted our paper, "Simple Dataset for Proof Method Recommendation in Isabelle/HOL (Dataset Description) [2]", where we introduced our dataset used for proof method recommendation in Isabelle/HOL [3]. This dataset is publicly available at Zenodo [1].

In this Appendix, we expound the feature extractor, we used to build the dataset by converting each invocation of proof method into a sequence of boolean values. The feature extractor consists of 113 assertions, and each assertion checks a certain property about the proof state at hand: an assertion returns true if the proof state satisfies the property under consideration, while the assertion returns false if it does not satisfy the property.

2 Terminology

Before exploring the 113 assertion, we first define certain terminologies specific to Isabelle/HOL.

⋆ This work was supported by the European Regional Development Fund under the project AI & Reasoning (reg. no.CZ.02.1.01/0.0/0.0/15_003/0000466) and by NII under NII-Internship Program 2019-2nd call.

- A *proof goal* is a conjecture being proved in Isabelle/HOL.
- A *proof method* is a sub-tool of Isabelle, with which human engineers guide Isabelle to prove a goal. Proof methods behave similarly to *tactics* in other LCF-style interactive theorem provers.
- A *keyword* is a command in Isabelle's proof language, Isar [4].
- A *theory (file)* is an Isabelle file, which contains definitions and theorems.
- A *proof context* is a background, based on which Isabelle users formulate conjectures and prove them.
- *Main* is the theory file that is often used as the standard library.
- *Pure* is the meta-logic of Isabelle.
- *HOL* is the classical higher-order logic of Isabelle/HOL defined in Pure.

3 The List of 113 Assertions

The following is the list of explanations to the list of assertions we used to construct the dataset. Some assertions were developed with certain target proof methods in mind, others were developed without clear intuitions. For those assertions developed with certain target methods in mind, we explained not only what they check but also why we developed them.

1. This assertion checks the proof state involves locally introduced assumptions.
2. This assertion checks if the proof goal involves a case distinction.
   This assertion may be useful to recommend the use of `case_tac` or `cases`.
3. This assertion checks if the first sub-goal involves any of the following constants:
   - `Bits.bits_class.test_bit`,
   - `Bits.bits_class.lsb`,
   - `Bits.bits_class.set_bit`,
   - `Bits.bits_class.bits`,
   - `Bits.bits_class.shiftl`, or
   - `Bits.bits_class.shiftr`
4. This assertion checks if the first sub-goal involves any of the following constants:
   - `Orderings.ord_class.less_eq`,
   - `Orderings.ord_class.less`, or
   - `Groups.plus_class.plus."`.
5. This assertion checks if the first sub-goal involves any constant that has a related `.pinduct` rule in the proof context.
   This assertion may be useful to recommend proof methods that are good at proving goals involving constants defined by the `function` keyword.
6. This assertion checks if the first sub-goal involves any constant that has a related `.psimp` rule in the proof context.
   This assertion may be useful to recommend proof methods that are good at proving goals involving constants defined by the `function` keyword.
7. This assertion checks if the first sub-goal involves any constant that has a related `.pelims` rule in the proof context.

This assertion may be useful to recommend proof methods that are good at proving goals involving constants defined by the `function` keyword.

8. This assertion checks if the first sub-goal involves any constant that has a related `.cases` rule in the proof context.
   This assertion may be useful to recommend proof methods that are good at proving goals involving constants defined by the `fun` or `function` keyword.
9. This assertion checks if the first sub-goal involves any constant that has a related `.intros` rule in the proof context.
   This assertion may be useful to recommend proof methods that are good at proving goals involving constants defined by the `inductive` or `inductive_set` keyword.
10. This assertion checks if the first sub-goal involves any constant that has a recursive simplification rule in the proof context.
    This assertion may be useful to recommend `induct`, `induction`, or `induct_tac`.
11. This assertion checks if the first sub-goal involves any constant defined in the `Num` theory.
12. This assertion checks if the first sub-goal involves any constant that has any of the following related rules in the proof context:
    - the `.abs_eq` rule,
    - the `.rsp` rule,
    - the `.transfer` rule, or
    - the `.rep_eq` rule.

    This assertion may be useful to recommend proof methods that are good at proving goals involving constants defined by the `lift_definition` keyword.
13. This assertion checks if the first sub-goal involves any constant that has any of the following related rules in the proof context:
    - the `.code` rule,
    - the `.ctr` rule, and
    - the `.sel` rule.

    This assertion may be useful to recommend proof methods that are good at proving goals involving constants defined by the `primcorec` keyword.
14. This assertion checks if the first sub-goal involves any constant that has a related `.axiom` rule in the proof context.
    This assertion may be useful to recommend proof methods that are good at proving goals involving constants introduced by the `locale` keyword.
15. This assertion attempts to check if the first sub-goal has a function that is not fully applied.
16. This assertion checks if the first sub-goal involves `BNF_Def.rel_fun`.
17. This assertion checks if the first sub-goal involves `Fun.map_fun`.
18. This assertion checks if the first sub-goal involves a schematic variable.
19. This assertion checks if the first sub-goal involves a constant defined in the `Real` theory.
20. This assertion checks if the first sub-goal involves a constant defined in the `List` theory.

21. This assertion checks if all constants appearing in the first sub-goal are already defined in the `Main`.
22. This assertion checks if the first sub-goal is classified as “equational" problem or not.
23. This assertion checks if the first sub-goal is classified as “inductive" problem or not.
24. This assertion checks if the first sub-goal is classified as “co-inductive" problem or not.
25. This assertion checks if the first sub-goal is classified as “unkown" problem or not.
26. This assertion checks if the outermost constant of the first sub-goal is the Pure equation.
27. This assertion checks if the outermost constant of the first sub-goal is the HOL equation.
28. This assertion checks if the outermost constant of the first sub-goal is the Pure implication.
29. This assertion checks if the outermost constant of the first sub-goal is the HOL implication.
30. This assertion checks if the outermost constant of the first sub-goal is the Pure universal quantifier.
31. This assertion checks if the outermost constant of the first sub-goal is the HOL universal quantifier.
32. This assertion checks if the outermost constant of the first sub-goal is the HOL existential quantifier.
33. This assertion checks if the first sub-goal involves the HOL existential quantifier, but not as the outermost constant.
34. This assertion checks if the first sub-goal involves the HOL equation, but not as the outermost constant.
35. This assertion checks if the first sub-goal involves the Pure implication, but not as the outermost constant.
36. This assertion checks if the first sub-goal involves the HOL implication, but not as the outermost constant.
37. This assertion checks if the first sub-goal involves the Pure universal quantifier, but not as the outermost constant.
38. This assertion checks if the first sub-goal involves the HOL universal quantifier, but not as the outermost constant.
39. This assertion checks if the first sub-goal involves the HOL existential quantifier, but not as the outermost constant.
40. This assertion checks if the first sub-goal involves `All` and `_dom` when printed as a string.
    This assertion may be useful to recommend proof methods that are good at proving goals function termination.
41. This assertion checks if the first sub-goal involves `_sumC` when printed as a string.
    This assertion may be useful to recommend proof methods that are good at proving goals involving constants defined by the `function` keyword.

42. This assertion checks if the first sub-goal involves `OFCLASS` when printed as a string.
    This assertion may be useful to recommend proof methods that are good at instantiation proofs.
43. This assertion checks if the first sub-goal involves `local.` and `_axioms` when printed as a string.
    This assertion may be useful to recommend `intro_locales` and `unfold_locales`.
44. This assertion checks if the outermost constant of the first sub-goal has all the following related rules in the proof context:
    - the `_def` rule, and
    - the `_axioms_def` rule.

    This assertion may be useful to recommend `unfold_locales`.
45. This assertion checks if the first sub-goal has any terms that have the type `Word.word`.
46. This assertion checks if the outermost constant of the first sub-goal is the HOL conjunction.
47. This assertion checks if the outermost constant of the first sub-goal is the HOL disjunction.
48. This assertion checks if the outermost constant of the first sub-goal is the HOL negation.
49. This assertion checks if the first sub-goal involves any constant defined in the `Set` theory.
50. This assertion checks if the first sub-goal involves any constant that has a related `.induct` rule in the proof context.
    This assertion may be useful to recommend proof methods that are good at proving goals involving constants defined by the `inductive` or `inductive_set` keyword.
51. This assertion checks if the first sub-goal involves any constant defined in the `Nat` theory.
52. This assertion checks if the first sub-goal involves any constant defined in the `Int` theory.
53. This assertion checks if the first sub-goal involves a term that has a rule that contains all of the following strings in its name in the proof context:
    - `rec_transfer`.

    This assertion may be useful to recommend proof methods that are good at proving goals involving a term of a type defined by the `datatype` keyword.
54. This assertion checks if the first sub-goal involves a term that has a rule that contains all of the following strings in its name in the proof context:
    - `inj_map_strong`.

    This assertion may be useful to recommend proof methods that are good at proving goals involving a term of a type defined by the `datatype` keyword.
55. This assertion checks if the first sub-goal involves a term that has a rule that contains all of the following strings in its name in the proof context:
    - `Abs_`, and
    - `_ext_inject`.

This assertion may be useful to recommend proof methods that are good at proving goals involving a term of a record type.

56. This assertion checks if the first sub-goal involves `Filter.eventually`.
    This assertion may be useful to recommend `eventually_elim`.
57. This assertion checks if the first sub-goal involves any constant that has a related `.inducts` rule in the proof context.
    This assertion may be useful to recommend proof methods that are good at proving goals involving constants defined by the `inductive` keyword.
58. This assertion checks if the first sub-goal involves any constant that has a related `.elims` rule in the proof context.
    This assertion may be useful to recommend proof methods that are good at proving goals involving constants defined by the `fun` keyword.
59. This assertion checks if the outermost constant of the first sub-goal is `Abstract.temp_strengthening`.
60. This assertion checks if the first sub-goal involves `Fun.map_fun`.
    This assertion may be useful to recommend `mkek_induct`.
61. This assertion checks if the first sub-goal involves a term that has a rule that contains all of the following strings in its name in the proof context:
    - `corec_disc`.

    This assertion may be useful to recommend proof methods that are good at proving goals involving a term of a type defined by the `codatatype` keyword.
62. This assertion checks if the first sub-goal involves any of the following constants:
    - `BNF_Def.rel_fun`, and
    - `Fun.map_fun`.
63. This assertion checks if the first sub-goal's term size is larger than 5.
64. This assertion checks if the first sub-goal's term size is larger than 10.
65. This assertion checks if the first sub-goal's term size is larger than 20.
66. This assertion checks if the first sub-goal's term size is larger than 40.
67. This assertion checks if the first sub-goal's term size is larger than 80.
68. This assertion checks if the first sub-goal involves the following constant:
    - `Groups.times_class.times`.

    This assertion may be useful to recommend `algebra`.
69. This assertion checks if the first sub-goal involves the following constant:
    - `Groups.plus_class.plus`.

    This assertion may be useful to recommend `algebra`.
70. This assertion checks if the first sub-goal involves the following constant:
    - `Power.power_class.power`.

    This assertion may be useful to recommend `algebra`.
71. This assertion checks if the first sub-goal involves the following constant:
    - `Groups.minus_class.minus`.

    This assertion may be useful to recommend `algebra`.
72. This assertion checks if the first sub-goal involves the following constant:
    - `Groups.uminus_class.uminus`.

    This assertion may be useful to recommend `algebra`.
73. This assertion checks if the first sub-goal involves the following constant:

- `Fields.inverse_class.inverse_divide`.

This assertion may be useful to recommend `algebra`.

74. This assertion checks if the first sub-goal involves any of the following constants:
    - `Groups.times_class.times`,
    - `Groups.plus_class.plus`,
    - `Power.power_class.power`,
    - `Groups.minus_class.minus`,
    - `Groups.uminus_class.uminus`, and
    - `Fields.inverse_class.inverse_divide`.

    This assertion may be useful to recommend `algebra`.
75. This assertion checks if the first sub-goal involves any constant defined in the `Group` theory.
    This assertion may be useful to recommend `algebra`.
76. This assertion checks if the first sub-goal involves any constant defined in the `Language` theory.
    This assertion may be useful to recommend `vcg`.
77. This assertion checks if the first sub-goal involves any constant defined in the `Cartesian_Euclidean_Space` theory.
78. This assertion checks if the first sub-goal involves `vec` when printed as a string.
    This assertion may be useful to recommend `vector`.
79. This assertion checks if the first sub-goal has any terms that have the type `Finite_Cartesian_Product.vec`.
80. This assertion checks if the first sub-goal involves more than one occurrences of `Pure.imp`.
    This assertion may be useful to recommend `hypsubst`.
81. This assertion checks if the first sub-goal involves more than two occurrences of `Pure.imp`.
    This assertion may be useful to recommend `hypsubst`.
82. This assertion checks if the first sub-goal involves any constant that has a related `.unfold` rule in the proof context.
    This assertion may be useful to recommend proof methods that are good at proving goals involving constants defined by the `fixrec` keyword.
83. This assertion checks if the first sub-goal involves a term that has a rule that contains all of the following strings in its name in the proof context:
    - `unfold`.

    This assertion may be useful to recommend proof methods that are good at proving goals involving a term of a type defined by the `domain` keyword.
84. This assertion checks if the first sub-goal satisfies any of the following conditions:
    - the first sub-goal involves any constant that has a related `.unfold` rule in the proof context.
    - the first sub-goal involves any term that has a related `.unfold` rule in the proof context.

85. This assertion checks if the first sub-goal involves a term that has a rule that contains all of the following strings in its name:
    - `narrowing`,
    - `simps`

    This assertion may be useful to recommend `nominal_induct`.
86. This assertion checks if the first sub-goal involves a term that has a rule that contains all of the following strings in its name:
    - `constr_rep`.

    This assertion may be useful to recommend `nominal_induct`.
87. This assertion checks if the first sub-goal involves a term that has a rule that contains all of the following strings in its name:
    - `rec_unique`.

    This assertion may be useful to recommend `nominal_induct`.
88. This assertion checks if the first sub-goal involves all of the following when printed as a string.
    - `thesis`,
    - $\Longrightarrow$ and,
    - `)`

    This assertion may be useful to recommend proof methods that are good at proving goals started by the `obtain` keyword.
89. This assertion checks if the first sub-goal involves a Pure implication that has an application of a constant that has a related `.inducts` rule in the proof context.
90. This assertion checks if the first sub-goal involves any of the following constants:
    - `Orderings.ord_class.less_eq`,
    - `Orderings.ord_class.less`,
    - `Orderings.greater`, and
    - `Orderings.greater_eq`.

    This assertion may be useful to recommend `linarith`.
91. This assertion checks if the first sub-goal has any terms that have any of the following types:
    - `Nat.nat`,
    - `Int.int`,
    - `Rat.rat`, and
    - `Real.real`.
92. This assertion checks if any of the following assertions return `true`:
    - assertion 90, and
    - assertion 91.
93. This assertion checks if the first sub-goal has any terms that have the type `Real.real`.
94. This assertion checks if the first sub-goal involves any constant that has a related `.rsp` rule in the proof context.
    This assertion may be useful to recommend proof methods that are good at proving goals involving constants defined by the `quotient_definition` keyword.

95. This assertion checks if the first sub-goal involves a term that has a rule that contains the following string in its name:
    - `rec_unique`.

    This assertion may be useful to recommend proof methods that are good at proving goals involving constants defined by the `quotient_definition` keyword.
96. This assertion checks if any of the following assertions return `true`:
    - assertion 94, and
    - assertion 95.
97. This assertion checks if both of the following assertions return `true`:
    - assertion 94, and
    - assertion 95.
98. This assertion checks if the first sub-goal involves a constant defined in the `Seq` theory.
99. This assertion checks if the first sub-goal involves a constant defined in the `Sequence` theory.
100. This assertion checks if any of the following assertions return `true`:
    - assertion 98, and
    - assertion 99.
101. This assertion checks if both of the following assertions return `true`:
    - assertion 98, and
    - assertion 99.
102. This assertion checks if the first sub-goal involves any terms that have a type defined in the `Seq` theory.
103. This assertion checks if the first sub-goal involves any terms that have a type defined in the `Sequence` theory.
104. This assertion checks if the first sub-goal involves any terms that have a type defined in any of the following theories:
    - the `Seq` theory, and
    - the `Sequence` theory.
105. This assertion checks if any of the following assertions return `true`:
    - assertion 100, and
    - assertion 104.
106. This assertion checks if the first sub-goal involves `OFCLASS` and `countable_class` when printed as a string.
107. This assertion checks if the first sub-goal has any terms that have the type `Sigma_Algebra.measure`.
108. This assertion checks if the first sub-goal involves any of the following constant:
    - `Borel_Space.borel_measurable`.
109. This assertion checks if any of the following assertions return `true`:
    - assertion 107, and
    - assertion 108.
110. This assertion checks if both of the following assertions return `true`:
    - assertion 107, and
    - assertion 108.

111. This assertion checks if the first sub-goal involves any of the following constant:
    - `Archimedean_Field.floor_ceiling_class.floor`.

    This assertion may be useful to recommend `mir`.
112. This assertion checks if the first sub-goal involves any constant defined in the `Hoare` theory.
    This assertion may be useful to recommend `hoare`.
113. This assertion checks if the first sub-goal involves `COBEGIN` when printed as a string.
    This assertion may be useful to recommend `oghoare`.

References

1. Nagashima, Y.: Simple Dataset for Proof Method Recommendation in Isabelle/HOL (May 2020). https://doi.org/10.5281/zenodo.3819026, `https://doi.org/10.5281/zenodo.3819026`
2. Nagashima, Y.: Simple dataset for proof method recommendation in isabelle/hol (dataset description). CoRR **abs/2004.10667** (2020), `https://arxiv.org/abs/2004.10667`
3. Nagashima, Y., He, Y.: PaMpeR: proof method recommendation system for isabelle/hol. In: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE 2018, Montpellier, France, September 3-7, 2018. pp. 362–372 (2018), `https://doi.org/10.1145/3238147.3238210`
4. Wenzel, M.: Isabelle, Isar - a versatile environment for human readable formal proof documents. Ph.D. thesis, Technical University Munich, Germany (2002), `http://tumb1.biblio.tu-muenchen.de/publ/diss/in/2002/wenzel.pdf`