Санкт-Петербургский государственный университет

Санкт-Петербургский государственный университет

Кафедра системного программирования

Формализация теории определимости в Lean

Илья Дудников

Научный руководитель: к.ф.-м.н. М.Р. Старчак, ассистент кафедры информатики

Санкт-Петербург
2023

Введение

- Теория определимости позволяет классифицировать объекты, узнавать их свойства

Введение

- Теория определимости позволяет классифицировать объекты, узнавать их свойства
- Доказательства технически непростые и могут содержать ошибки

Введение

- Теория определимости позволяет классифицировать объекты, узнавать их свойства
- Доказательства технически непростые и могут содержать ошибки
  - Нужны убедительные средства

igl2020 и mathlib

- **igl2020** — проект иллинойских студентов, целью которого была формализация **теории моделей** в Lean.

igl2020 и mathlib

- **igl2020** — проект иллинойских студентов, целью которого была формализация **теории моделей** в Lean.
- **mathlib** — это библиотека для Lean, поддерживаемая пользователями.

igl2020 и mathlib

- **igl2020** — проект иллинойских студентов, целью которого была формализация **теории моделей** в Lean.
- **mathlib** — это библиотека для Lean, поддерживаемая пользователями.

igl2020

- Термы
- Формулы
- Языки
- Структуры

igl2020 и mathlib

- **igl2020** — проект иллинойских студентов, целью которого была формализация **теории моделей** в Lean.
- **mathlib** — это библиотека для Lean, поддерживаемая пользователями.

igl2020

- Термы
- Формулы
- Языки
- Структуры

mathlib

- Языки
- Структуры
- Satisfiability
- Definability of sets

Постановка задачи

Целью работы является формализация некоторых понятий определимости в Lean.
Задачи:

- Формализовать понятие арифметических языков и арифметических структур
- Доказать корректность введенных формализаций
- Ввести понятие определимости предикатов в арифметических структурах
- Доказать теорему о сохранении определимости в арифметических структурах

Арифметические структуры и языки

В igl2020 языки заданы тройкой (F, R, C), структуры — тройкой, задающей интерпретацию.

Арифметические структуры и языки

В igl2020 языки заданы тройкой (F, R, C), структуры — тройкой, задающей интерпретацию.
Арифметические структуры **проще**.

Арифметические структуры и языки

В igl2020 языки заданы тройкой (F, R, C), структуры — тройкой, задающей интерпретацию.
Арифметические структуры **проще**. Формулы арифметических языков можно интерпретировать с помощью $N\_arith\_semiring = \langle \mathbb{N}, 0, 1, +, *, < \rangle$.

Арифметические структуры и языки

В igl2020 языки заданы тройкой (F, R, C), структуры — тройкой, задающей интерпретацию.
Арифметические структуры **проще**. Формулы арифметических языков можно интерпретировать с помощью $N\_arith\_semiring = \langle \mathbb{N}, 0, 1, +, *, < \rangle$.

```
structure arith_lang : Type 1 :=
  (n : ℕ+)
  (ar : fin n → ℕ+)
```

Арифметические структуры и языки

В igl2020 языки заданы тройкой (F, R, C), структуры — тройкой, задающей интерпретацию.
Арифметические структуры **проще**. Формулы арифметических языков можно интерпретировать с помощью $N\_arith\_semiring = \langle \mathbb{N}, 0, 1, +, *, < \rangle$.

```
structure arith_lang : Type 1 :=
  (n : ℕ+)
  (ar : fin n → ℕ+)
```

```
structure arith_struc (L : arith_lang) :=
  (rels : vector (formula
    ordered_semiring_lang) L.n)
  (ar_proof : ∀i,
    formula.count_free_vars_list
    (rels.nth i) = L.ar i)
```

`rels` — набор базовых отношений, `ar_proof` — сертификат, подтверждающий, что арность i-го отношения равна $L.ar(i)$.

Арифметические структуры и языки

Для доказательства корректности введенных определений, мы показываем, что `arith_lang` является `lang`, а `arith_struc` — `struc`.

Арифметические структуры и языки

Для доказательства корректности введенных определений, мы показываем, что `arith_lang` является `lang`, а `arith_struc` — `struc`.

- Любой `arith_lang` является языком, в котором нет констант и функциональных символов и есть n отношений, арность которых задается ar

Арифметические структуры и языки

Для доказательства корректности введенных определений, мы показываем, что `arith_lang` является `lang`, а `arith_struc` — `struc`.

- Любой `arith_lang` является языком, в котором нет констант и функциональных символов и есть n отношений, арность которых задается ar
- `arith_struc` — структура, её носитель — $\mathbb{N}$, констант и функциональных символов нет. Отношения задаются:

$$v \in S.R\ i \Leftrightarrow v \models \varphi$$

где φ — формула, задающая отношение.

Определимость в арифметических структурах

Арифметические структуры решают проблему.

$$\mathrm{Def}(p, M) := \exists \varphi : \text{formula } L\ (\forall va : \mathbb{N} \to \mathbb{N}\ (va \models p \Leftrightarrow va \models \varphi))$$

$$\mathrm{Def}(M) := \{\psi \mid \mathrm{Def}(\psi, M)\}$$

Определимость в арифметических структурах

Арифметические структуры решают проблему.

$$\mathrm{Def}(p, M) := \exists \varphi : \text{formula } L \ (\forall va : \mathbb{N} \to \mathbb{N} \ (va \models p \Leftrightarrow va \models \varphi))$$

$$\mathrm{Def}(M) := \{\psi \mid \mathrm{Def}(\psi, M)\}$$

```
def predicate_is_definable_in_arith_struc
  {L₁ : arith_lang} (S : arith_struc L₁)
  (pred : formula ordered_semiring_lang)
    :=
  ∃ φ : formula L₁,
    ∀ va : ℕ → ℕ, va ⊨ φ ↔ va ⊨ pred
```

Определимость в арифметических структурах

Арифметические структуры решают проблему.

$$\mathrm{Def}(p, M) := \exists \varphi : \text{formula } L\ (\forall va : \mathbb{N} \to \mathbb{N}\ (va \models p \Leftrightarrow va \models \varphi))$$

$$\mathrm{Def}(M) := \{\psi \mid \mathrm{Def}(\psi, M)\}$$

```
def predicate_is_definable_in_arith_struc
  {L₁ : arith_lang} (S : arith_struc L₁)
  (pred : formula ordered_semiring_lang)
    :=
  ∃ φ : formula L₁,
    ∀ va : ℕ → ℕ, va ⊨ φ ↔ va ⊨ pred
```

```
def definable_predicates
  {L₁ : arith_lang} (S : arith_struc L₁)
    :=
  { φ |
    predicate_is_definable_in_arith_struc
    S φ }
```

Теорема о сохранении определимости

Теорема

Пусть S_1, S_2 – арифметические структуры, причем любой базовый предикат S_1 определим в S_2. Тогда множество всех предикатов, определимых в S_1 содержится в множестве всех предикатов, определимых в S_2. То есть,

$$(\forall p \in S_1.rels\ (p \in \mathrm{Def}(S_2))) \Rightarrow \mathrm{Def}(S_1) \subseteq \mathrm{Def}(S_2)$$

Теорема о сохранении определимости

Теорема

Пусть S_1, S_2 – арифметические структуры, причем любой базовый предикат S_1 определим в S_2. Тогда множество всех предикатов, определимых в S_1 содержится в множестве всех предикатов, определимых в S_2. То есть,

$$(\forall p \in S_1.rels\ (p \in \mathrm{Def}(S_2))) \Rightarrow \mathrm{Def}(S_1) \subseteq \mathrm{Def}(S_2)$$

```
theorem subset_if_predicates_definable {L₁ L₂ : arith_lang}
  (S₁ : arith_struc L₁) (S₂ : arith_struc L₂) :
  (∀ pred ∈ S₁.rels.to_list, predicate_is_definable_in_arith_struc S₂ pred) →
    definable_predicates S₁ ⊆ definable_predicates S₂ :=
```

Теорема о сохранении определимости

Теорема

Пусть S_1, S_2 – арифметические структуры, причем любой базовый предикат S_1 определим в S_2. Тогда множество всех предикатов, определимых в S_1 содержится в множестве всех предикатов, определимых в S_2. То есть,

$$(\forall p \in S_1.rels\ (p \in \mathrm{Def}(S_2))) \Rightarrow \mathrm{Def}(S_1) \subseteq \mathrm{Def}(S_2)$$

```
theorem subset_if_predicates_definable {L₁ L₂ : arith_lang}
  (S₁ : arith_struc L₁) (S₂ : arith_struc L₂) :
  (∀ pred ∈ S₁.rels.to_list, predicate_is_definable_in_arith_struc S₂ pred) →
    definable_predicates S₁ ⊆ definable_predicates S₂ :=
```

Доказательство

Теорема доказывается индукцией по ψ, где ψ — такая формула языка L_1, что $va \models \psi \Leftrightarrow va \models \varphi$ для любых $va : \mathbb{N} \to \mathbb{N}$ и некоторой формулы $\varphi \in \mathrm{Def}(S_1)$.

Результаты

- Введены понятия арифметических языков и арифметических структур
- Доказана корректность введенных формализаций
- Формализовано понятие определимости предикатов в арифметических структурах
- Доказана теорема о сохранении определимости в арифметических структурах

Problems in the literature. Presburger arithmetic is the first-order theory of $\langle \mathbb{Z}, 0, 1, +, < \rangle$ Presburger arithmetic with divisibility (PAD) is the extension of PA obtained when we add a binary divisibility predicate

...

Their procedure does not eliminate all quantifiers but rather yields a sentence in the $\prod_0$-fragment of PAD. (Decidability of the BIL language would then follow from the result of Lipshitz [26].) Then, they briefly argue how the algorithm generalizes to $\forall\exists_R \mathrm{PAD}^+$. There are two crucial problems in the argument from [6] that we have summarized here (and which were reproduced in Lechner's work): First, the quantifier-elimination procedure of Bozga and Iosif does not directly work for BIL. Indeed, not all BIL sentences satisfy the conditions required for the CRT to be applicable as used in their algorithm. Second, there is no way to generalize their algorithm to $\forall\exists_R \mathrm{PAD}^+$ since the language is undecidable.

— Revisiting Parameter Synthesis for One-Counter Automata, Guillermo A. Pérez, Ritam Raha, 2022

Корректность арифметичесих структур и языков

```
def arith_lang.to_lang (L :
    arith_lang) : lang :=
  {
    F := λ _, empty,
    R := λ x, let ar_vec :=
    vector.of_fn L.ar in
              fin
    (ar_vec.to_list.count x),
    C := empty
  }
```

```
def arith_struc.to_struc {L : arith_lang} (S :
    arith_struc L) : struc L :=
  {
    R := λ x r,
      if ((list.of_fn L.ar).count x) = 0 then ∅
      else
        let index := -- index of the desired
    formula
        let φ := S.rels.nth index in
        { v : vector ℕ x |
          let va : ℕ → N_arith_semiring.univ :=
    λ var,
            -- constructing variable assignment
    map
          in va ⊨ φ }
  }
```